M3AAWG Companion Document: Recipes for Encrypting DNS Stub Resolver-to-Recursive Resolver Traffic

Version 1.0
September 2019

The direct URL to this paper is: www.m3aawg.org/dns-crypto-recipes

This document is intended to accompany and complement the companion document, “M3AAWG Tutorial on Third Party Recursive Resolvers and Encrypting DNS Stub Resolver-to-Recursive Resolver Traffic” (www.m3aawg.org/dns-crypto-tutorial).

This document was produced by the M3AAWG Data and Identity Protection Committee.

Table of Contents

Introduction ................................................................................................................................................................... 2

Section 1. Configuring a Mac OS X System to Encrypt Stub-to-Recursive-Resolver Traffic.......................... 3
Example A: Implementing DNS over HTTPS Using the Cloudflared-proxy to Cloudflare's 1.1.1.1 ............... 3
Example B: Implementing DNSCrypt Using DNSCrypt-proxy 2 with Quad9's DNSSEC-enabled and Filtered Service ............................................................................................................................................................................ 6
Example C: Implementing DNS over TLS Using Stubby with the Stubby Author's DNS over TLS Test Servers ............................................................................................................................................................................... 8
Example D: Implementing DNS over HTTPS for the Firefox Web Browser Only............................................. 10

Section 2. Configuring Windows 10 to Encrypt Stub-to-Recursive-Resolver Traffic ........................................ 12

Section 3. Configuring an iPhone to Encrypt Stub-to-Recursive-Resolver Traffic ....................................... 19
Example A: Using the Cloudflare 1.1.1.1 Native Application ........................................................................ 19

Example B: Using DNSCloak ................................................................................................................................. 21

Section 4. Configuring an Android Phone to Encrypt Stub-to-Recursive-Resolver Traffic ........................... 24

Section 5. Configuring a Raspberry Pi Running Raspbian Stretch to Run Unbound and DNS over TLS ........ 27


Section 7. Diagnosing and Testing DNS Stub-to-Recursive Resolver Traffic ................................................ 37

Conclusion ......................................................................................................................................................... 39
Introduction

Before deciding to use any encrypted recursive-resolver protocol or service, a key point to understand is that encrypting stub-to-recursive traffic is still quite experimental. If you try encrypting your stub-to-recursive traffic, you will be doing so as an early adopter or a pilot project.

This means:

- Standards are still under development and implementations likely will evolve while you are using the technology. There may be bugs or other issues that still need to be fixed.
- Your operating system likely will not have native support for encrypted recursive resolvers, so you will need to add a third party package to try encrypting stub-to-recursive-resolver traffic.
- Your ISP will likely not have encrypted recursive resolvers for you to use, so you will need to trust a third party with your DNS queries.
- Paradoxically, using an encrypted recursive resolver will likely make your traffic stand out because you are doing something that is unusual.
- Since the protection that encrypted recursive resolvers deliver is still imperfect, you increase the chance that your traffic will actually end up being monitored.

If you are interested in encrypted recursive resolvers, try at least one option for your own personal traffic. It is one thing to read about these services and another to actually try using one (or more) of them yourself.

To facilitate this, we have developed detailed "recipes" for various protocols/providers for most common platforms:

- Mac OS X
- MS Windows 10
- iPhone
- Android
- Raspbian on a Raspberry Pi

The recipes in this companion document should be sufficient to allow you to set up the system and be running with at least one encrypted recursive resolver solution for each popular platform.

One remaining question you may have: Should you encrypt device-by-device or at the home gateway only?

If you are like most people, you have multiple devices – maybe a desktop, a laptop, a smart phone, a tablet or e-reader, a smart TV, and more. You probably also use at least some of those devices at multiple locations – home, office, restaurants, your children’s schools, the gym, while you are traveling, and so on. Do you plan to encrypt the DNS traffic to/from each of those devices on a device-by-device basis? (It can be difficult to encrypt multiple devices one by one.) Or are you going to encrypt all of your DNS traffic at your home gateway (i.e., your broadband router)? If you decide to go this route, what will you do when you are using a third party network (e.g., at the office, at restaurants, at your children’s schools, the gym, while traveling, or somewhere else)?

**The best option if you are going to encrypt DNS: Do both – per-device DNS encryption and at-the-gateway DNS encryption by default**
Section 1. Configuring a Mac OS X System to Encrypt Stub-to-Recursive-Resolver Traffic

<table>
<thead>
<tr>
<th>Example</th>
<th>Encryption Technology</th>
<th>Software</th>
<th>Recursive Resolver Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>DNS over HTTPS</td>
<td>Cloudflared-proxy</td>
<td>Cloudflare’s 1.1.1.1</td>
</tr>
<tr>
<td>B</td>
<td>DNSCrypt</td>
<td>DNSCrypt-proxy 2</td>
<td>Quad9’s 9.9.9.9</td>
</tr>
<tr>
<td>C</td>
<td>DNS over TLS</td>
<td>Stubby</td>
<td>The Stubby author’s test</td>
</tr>
<tr>
<td>D</td>
<td>DNS over HTTPS</td>
<td>Firefox TRR</td>
<td>Four different options shown</td>
</tr>
</tbody>
</table>

Example A: Implementing DNS over HTTPS Using the Cloudflared-proxy to Cloudflare’s 1.1.1

We will install cloudflared-proxy as described at: https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy/

The cloudflared-proxy can be installed on the Mac using the third-party Mac package manager “Homebrew” (see https://brew.sh/).

To install cloudflared-proxy for testing:

- As always, before any system maintenance, ensure that you have a recent backup of your system and that your system is fully patched with no pending updates.

- Now open a Terminal window and install Homebrew, if you have not previously done so. (As described above, do this at https://brew.sh/)

- Next, use Homebrew to install cloudflared:
  
  $ brew install cloudflare/cloudflare/cloudflared

  ![Brew Install Cloudflared]

- You may want to check to ensure that cloudflared is now actually installed:
  
  $ cloudflared --version

  You should see something that looks like:

  ![Cloudflared Version]

- Now run cloudflared (you will need your admin password because the application binds to a privileged port number):
  
  $ sudo cloudflared proxy-dns
After you have successfully entered your admin password, you should see something that looks like:

```
$ sudo cloudflared proxy-dns
Password:
INFO[0000] Adding DNS upstream
s-query
INFO[0000] Starting metrics server
INFO[0000] Adding DNS upstream
s-query
INFO[0000] Starting DNS over HTTPS proxy server
```

- In another terminal window, confirm that you have a recursive resolver answering at 127.0.0.1:

```
$ dig +short @127.0.0.1 cloudflare.com
198.41.214.162
198.41.215.162
```

- Now go to Mac OS System Preferences → System Preferences → Network

![Network Preferences](image)

- Select the relevant network interface (probably "WiFi"). Click on the Advanced button. Then click on the DNS tab.

```
```

**Important:** Carefully write down the current domain name server IP(s) you see!
• Once you have written down the current name server IP(s), change the DNS server to point at 127.0.0.1 instead. Normally you can do this by clicking the "+" sign underneath the left column and entering 127.0.0.1 as the new nameserver to use. Click OK to close the Advanced window and Apply to save the Network setting. Close the Network settings window.

• Do whatever other testing you would like to do. For example, use a browser to visit https://1.1.1.1/help

If you want to uninstall cloudflared after you are done testing it:

• Click on the Terminal window running cloudflared proxy-dns, then hit control-C to kill that process.

• Go to Mac OS System Preferences → Network → Advanced → DNS and change 127.0.0.1 back to the IP address(es) you were originally using.

• Click OK to close the Advanced window and click Apply to save the Network setting.

• Use Homebrew to remove the cloudflared software:
  $ brew remove cloudflare/cloudflare/cloudflared

Or, if you want to make cloudflared persistent; e.g., keep and use cloudflared from this point forward:

• Complete steps 5 and 6 as described at https://developers.cloudflare.com/1.1.1.1/dns-over-https/cloudflared-proxy/
Example B: Implementing DNSCrypt Using DNSCrypt-proxy 2 with Quad9's DNSSEC-enabled and Filtered Service

We will install DNSCrypt-proxy approximately as described at https://github.com/jedisct1/DNSCrypt-proxy/wiki/Installation-macOS

To install DNSCrypt-proxy 2 for testing:

- As always, before any system maintenance, ensure that you have a recent backup of your system, and that your system is fully patched with no pending updates.

- Now open a Terminal window and install Homebrew if you have not previously done so. Do this as described at https://brew.sh/

- Use Homebrew to install `wget` if you do not already have it:
  
  ```
  $ brew install wget
  ```

- Also use Homebrew to install `minisign` if you do not already also have that package:
  
  ```
  $ brew install minisign
  ```

- Get the DNSCrypt-proxy 2 code (Note: the version of this may change over time):
  
  ```
  $ wget https://github.com/jedisct1/DNSCrypt-proxy/releases/download/2.0.17/DNSCrypt-proxy-macos-2.0.17.tar.gz
  ```

- Also get the corresponding signature for verification:
  
  ```
  $ wget https://github.com/jedisct1/DNSCrypt-proxy/releases/download/2.0.17/DNSCrypt-proxy-macos-2.0.17.tar.gz.minisig
  ```

- Use minisign to confirm the downloaded code is intact and authentic:
  
  ```
  $ minisign -Vm DNSCrypt-proxy-*.tar.gz
  -P RWTk1xXqcTODeYttYMCMLo0YJHaFEHn7a3akqH1b/7QvIQXHPxKbJB5
  ```

  You should see something like: "Signature and comment signature verified"

- Assuming the signature verifies correctly, untar/ungzip that file by entering:
  
  ```
  $ tar xfv DNSCrypt-proxy-macos-2.0.17.tar.gz
  ```

  As that archive is processed, you will see a number of file names displayed.

- Change down into the directory just created by tar:
  
  ```
  $ cd macos
  ```

- Copy the template config file to DNSCrypt-proxy.toml:
  
  ```
  $ cp example-DNSCrypt-proxy.toml DNSCrypt-proxy.toml
  ```
• Tweak the options in that config file appropriately:
  • For example, tell DNSCrypt-proxy to **only** use DNSCrypt. To help ensure this, edit the `DNSCrypt-proxy.toml` file with your favorite text editor and make sure `DNSCrypt_servers = true` and `doh_servers = false`
  • If your connectivity is IPv6 enabled, try setting `ipv6_servers = true` and set `server_names = ['quad9-DNSCrypt-ip4-filter-pri', 'quad9-DNSCrypt-ip4-filter-alt', 'quad9-DNSCrypt-ip6-filter-pri', 'quad9-DNSCrypt-ip6-filter-alt']`
  • If you do **not** have IPv6, leave `ipv6_servers = false` and set `server_names = ['quad9-DNSCrypt-ip4-filter-pri', 'quad9-DNSCrypt-ip4-filter-alt']`
  • Save these changes and exit.

• Now run DNSCrypt-proxy as administrator:
  `$ sudo ./DNSCrypt-proxy`

After supplying your administrator password, you should see multiple lines of output, ending with something like:

```
[2018-11-05 15:25:28] [NOTICE] Server with the lowest initial latency: quad9-DNSCrypt-ip4-filter-alt (rtt: 20ms)
[2018-11-05 15:25:28] [NOTICE] DNSCrypt-proxy is ready - live servers: 4
```

**Note:** To learn more about the configuration of any particular server, see [https://DNSCrypt.info/public-servers](https://DNSCrypt.info/public-servers)

• Now go to *Mac OS System Preferences → Network*
  • Select the relevant network interface (likely "WiFi"). Click on the Advanced button. Then click on the DNS tab.
  • **Important:** Carefully write down the current domain name server IP(s) you see!
  • Once you have written down the current name server IP(s), change the DNS server to point at `127.0.0.1` instead.
  • Click OK to close the Advanced window, then click Apply to save the Network setting.

• Check to make sure the proxy works. Open a new terminal window and enter:
  `$ dig +short @127.0.0.1 example.com`

You should get one or more IP addresses returned. If the program is not working correctly, you will get no response.

• Perform whatever other testing you would like to do.
If you want to **uninstall DNSCrypt-proxy** after you are done testing it:

- Click on the Terminal window running DNSCrypt-proxy, then hit `control-C` to kill that process.

- Go to `Mac OS System Preferences → Network → Advanced → DNS` and change `127.0.0.1` back to the IP address(es) you were originally using.

- Click `OK` to close the `Advanced` window, then click `Apply` to save the Network setting.

- Delete the DNSCrypt-proxy software by dragging it to the Trash icon in Finder and then emptying the Trash.

Or, if you want to make **DNSCrypt-proxy persistent** (e.g., keep and use DNSCrypt-proxy from this point forward):

- Complete step 5 as shown at https://github.com/jedisct1/DNSCrypt-proxy/wiki/Installation-macos

**Example C: Implementing DNS over TLS Using Stubby with the Stubby Author's DNS over TLS Test Servers**

We will install Stubby approximately as described at https://dnsprivacy.org/wiki/pages/viewpage.action?pageId=3145812

By default, this installation uses the Stubby DNS over TLS Test Servers, as described at https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+Test+Servers#DNSPrivacyTestServers-DoTservers

**To install Stubby for testing:**

- As always, before any system maintenance, ensure that you have a recent backup of your system and that your system is fully patched with no updates pending.

- Now open a Terminal window and install Homebrew if you have not previously done so. Do this as described at https://brew.sh/

- Next, use Homebrew to install Stubby:
  ```bash
  $ brew install stubby
  (Note: this will also install Unbound and getdns if they are not already installed)
  ```

- If you do not have IPv6 enabled, edit `/usr/local/etc/stubby/stubby.yml` with your favorite editor and comment out the IPv6 upstream servers. (You can also look at some of the other commented-out public servers listed there.)

- Start Stubby for testing:
  ```bash
  $ sudo /usr/local/bin/stubby -C /usr/local/etc/stubby/stubby.yml
  ```
After entering your admin password, you should see something like:

[23:52:48.163152] STUBBY: Read config from file
/usr/local/etc/stubby/stubby.yml
[23:52:48.164771] STUBBY: DNSSEC Validation is ON
[23:52:48.164788] STUBBY: Transport list is:
(Authentication required)
[23:52:48.164801] STUBBY: (NOTE a Strict Profile only applies when TLS is the ONLY transport!!)
[23:52:48.164805] STUBBY: Starting DAEMON....

• Now go to Mac OS System Preferences → Network:
  - Select the relevant network interface (likely "WiFi"). Click on the Advanced button. Now click on the DNS tab.
  - Important: Carefully write down the current domain name server IP(s) you see!
  - Once you have written down your current name server IP(s), change the DNS server to point at 127.0.0.1 instead.
  - Click OK to close the Advanced window and Apply to save the Network setting.

• Check to make sure Stubby works. Open a new Terminal window and type:
  
  $ dig +short @127.0.0.1 example.com

You should get one or more IP addresses returned. If the program is not working correctly, you will get no response.

• Do whatever other testing you would like to do.

If you want to uninstall Stubby after you are done testing it:
• Click on the terminal window running Stubby, then hit control-C to kill that process.

• Go to Mac OS System Preferences → Network → Advanced → DNS and change 127.0.0.1 back to the IP address(es) you were originally using. Click OK to close the Advanced window, then click Apply to save the Network setting.

• Delete Stubby by typing:
  
  $ brew remove stubby

• If you do not have a reason to keep it, delete getdns by typing:
  
  $ brew remove getdns

• If you do not have a reason to keep it, you can also delete Unbound by entering:
  
  $ brew remove unbound

Or, if you want to make Stubby persistent (e.g., keep and use Stubby from this point forward):

  $ sudo brew services start stubby
Example D: Implementing DNS over HTTPS for the Firefox Web Browser Only

Examples A through C above illustrated installation of a DNS proxy enabling:
- DNS over HTTPS (Example A)
- DNSCrypt (Example B)
- DNS over TLS (Example C)

Use of a DNS proxy enables those protective protocols for all the applications that may be using DNS.

In this example, we will show enabling DNS over HTTPS just within current versions of the Firefox Web Browser. That is, by setting up TRR within Firefox, you will circumvent or override the current system recursive resolver setting.

Important: This only implements DNS over HTTPS for Firefox. All other applications on the Mac will still use your regular (non-encrypted) recursive resolvers.

To persistently enable DNS over HTTPS just for the Firefox Web browser:
- As always, before any system maintenance, ensure that you have a recent backup of your system and that your system is fully patched with no pending updates.
- Ensure that you have a current version of Firefox (63.0.1 or later should be fine. You can check the version of Firefox you have by going to Firefox → About Firefox).
- In the Firefox address bar, enter about:config (if asked, confirm that you do not mind voiding your warranty).
- In the search bar, enter trr to search for network.trr.mode
- Double-click on the resulting search item and set it to 2

The allowed values include:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Off (default). Use standard native resolver only (do not use TRR at all)</td>
</tr>
<tr>
<td>1</td>
<td>Race native against TRR. Use them both in parallel and go with the one that returns a result first.</td>
</tr>
<tr>
<td>2</td>
<td>TRR first. Use TRR first, and only if the name resolver fails, use the native resolver as a fallback.</td>
</tr>
<tr>
<td>3</td>
<td>TRR only. Only use TRR. Never use the native resolver (after the initial setup).</td>
</tr>
<tr>
<td>4</td>
<td>Shadow mode. Runs the TRR resolver in parallel with the native for timing and measurements but uses only the native resolver results.</td>
</tr>
<tr>
<td>5</td>
<td>Explicitly off. Also off, but selected off by choice and not default.</td>
</tr>
</tbody>
</table>

- By default you will be using the Cloudflare Firefox DNS over HTTPS server at https://mozilla.cloudflare-dns.com/dns-query
• If you would like to use a different DNS over HTTPS server, go to about:config and search for network.trr.uri

  • For Google's servers, set network.trr.uri to https://dns.google.com/resolve (see https://developers.google.com/speed/public-dns/docs/DNS over HTTPS)

  • For Quad9’s servers, set network.trr.uri to https://dns.quad9.net/dns-query (see https://www.quad9.net/doh-quad9-dns-servers)

  • For CleanBrowsing's security-filter server set, set network.trr.uri to https://doh.cleanbrowsing.org/doh/security-filter (see https://cleanbrowsing.org/dnsoverhttps)

Disabling DNS over HTTPS in Firefox TRR:

• In the Firefox address bar, enter about:config (If asked, confirm that you do not mind voiding your warranty.)

• In the search bar, enter trr to search for network.trr.mode, double-click on that item, and set it to 5
Section 2. Configuring Windows 10 to Encrypt Stub-to-Recursive-Resolver Traffic

While it is possible to install a traditional encrypted DNS proxies command line in Microsoft Windows, most Windows users will probably prefer a point-and-click graphical interface. Fortunately, Simple DNSCrypt\(^1\) is an excellent option for that scenario:

- Most Windows 10 users will want to click the "Download .msi (X64 Installer)" from the bottom of the Simple DNSCrypt page.

- After the file downloads, click on the downloaded file to begin installation, confirming that you want to open the executable and (if asked) confirm that you are okay with the installer modifying your system. You should see:

\(^1\) https://simplednscrypt.org/
• Click Next to proceed. You should see:

![Configure Shortcuts]

• Click Next.

• The next screen of the installer will ask you to select where you would like to install Simple DNSCrypt:

![Select Installation Folder]

• Click Next to proceed when you have selected the location where you would like to install the programs. You should see:

![Completing the Simple DNSCrypt x64 Setup Wizard]
- Click Finish to close the installer and launch Simple DNSCrypt. You will then be able to configure Simple DNSCrypt:

![Simple DNSCrypt configuration screen](image)

- There are two primary approaches you can take to configuring Simple DNSCrypt:
  
  - You can select the type of server you want and let Simple DNSCrypt pick the servers that meet those specifications. (In this case, the sample screen capture above shows selecting IPv4-accessible servers that support DNSSEC but that do not filter or log.)
  
  - Alternatively, you can go to the Resolvers panel and select a specific resolver you like, then click on Apply Settings.

![Simple DNSCrypt resolvers screen](image)

In the Resolver panel shown above, we have selected Cloudflare's 1.1.1.1/1.0.0.1 anycast resolver and then disabled "automatic" mode.
• The next screen shows the Advanced panel and the settings available from it:

In this case, we have enabled the DNS Cache and blocked IPv6 requests for improved performance. If we were using a VPN, we might want to tunnel all DNS traffic over TCP, but since we are not in this example, we left it unchecked. We are not planning to have this system act as a resolver for other devices either, so we left Global Resolver disabled. We accepted the default Quad9 Fallback Resolver to handle bootstrapping and captive portal situations.

• Click Apply Settings and close that interface.

• Important: Now go back to the Main Menu, instruct DNSCrypt (the service) to start and select your interface card:

We will instruct the Windows WiFi connection to route DNS requests via Simple DNSCrypt on the next page.

We are going to access Cloudflare's 1.1.1.1 via the forwarder that is running locally at 127.0.0.1 port 53.
• Begin by opening Windows Settings (the gear icon):

![Windows Settings](image1.png)

• Select Network & Internet from that panel. You should see the Status page. Select Change adapter options:

![Status page](image2.png)

• Select your network interface (usually Wireless Network Connection), then click Change settings of this connection

![Change settings of this connection](image3.png)
• Double-click on Internet Protocol Version 4 (TCP/IPv4):

![Image of Internet Protocol Version 4 (TCP/IPv4) properties]

• Now set the DNS server to manual and enter 127.0.0.1 as shown. Click OK.

![Image of DNS server settings]

• All that is left to do now is to confirm the settings are working as they should be. Open your Web browser and try going to https://1.1.1.1/help.

• If it is operating correctly, you should see something like the following:

![Image of debug information output]

M3AAWG Recipes for Encrypting DNS Stub-to-Recursive-Resolver Traffic
• Note that in this case, while we configured Simple DNSCrypt to **not** use IPv6, it actually is on the sample system if we were to want to take advantage of it.

• As one last test, restart your system. After restarting, revisit https://1.1.1.1/help and confirm that you are **still** seeing DNS via the configured service (as shown above).

• If so, you should be all set!

**Uninstalling Simple DNSCrypt**

Uninstalling Simple DNSCrypt requires three steps:

1. Our first step is to return to using the automatically recommended DNS servers from our ISP.

   To do this:
   a. Go to *Windows Settings* (the gear icon) → *Network & Internet* → *Change Adaptor Options*
   b. Pick your interface (normally the Wireless Network), then click *Changes Settings of This Connection*
   c. Double-click on *Internet Protocol Version 4 (TCP/IPv4).*
   d. Ensure that *Obtain DNS Server Address Automatically* is checked. If it is not, select it. Save the settings and exit.

2. Now remove the DNSCrypt system service. You can do this from the *Advanced Settings* menu tab on the Simple DNSCrypt interface:

3. Finally, uninstall the Simple DNSCrypt software from *Windows Settings* → *Apps and Features*:
Section 3. Configuring an iPhone to Encrypt Stub-to-Recursive-Resolver Traffic

Example A: Using the Cloudflare 1.1.1.1 Native Application

Cloudflare recently released their "1.1.1.1" app via the Apple App Store. (It is somewhat confusingly named because the app has the same name as the IP address of their recursive resolver service.) We tested it on an iPhone A1453 (iPhone 5S CDMA connecting via Sprint) and it worked well. The remainder of this section will show the process of setting up the 1.1.1.1 app.

Find the 1.1.1.1 app in the Apple App Store, download it and install it – it is free.
M3AAWG Recipes for Encrypting DNS Stub-to-Recursive-Resolver Traffic
Example B: Using DNSCloak

If you are looking for an alternative to the native Cloudflare 1.1.1.1 app, you can also try using DNSCloak.

The phone configured for this demonstration normally uses Sprint's own DNS servers, as can be seen by visiting https://1.1.1.1/help from Chrome:

![Debug Information](image)

For the purposes of this document, we will install and configure DNSCloak available via the App Store:
Open the application once it has downloaded and scroll down to select the provider you want to use – we will select Cloudflare's 1.1.1.1:

We will need to authorize installation of a VPN profile for the application to work:
Once a profile has been selected and installed, you will see a screen that looks like the screenshot below. (Note the orange square, which is the sign that DNSCloak is running. It will change to a triangle if it is off; if you see a triangle, click it to start the app.)

![Screen shot of DNSCloak app](image)

We can confirm use of the server by revisiting [https://1.1.1.1/help](https://1.1.1.1/help) and noting the new configuration that is reported.

![Debug Information](image)

Uninstalling DNSCloak from your iPhone:

To remove DNSCloak from your iPhone, simply delete the application as you normally would any other app:

- Go to the app on your home screen and hold your finger down on the DNSCloak app icon.
- When the app begins to shake, click the X to delete it.
- Confirm that you want to delete the app.

Your DNS will revert to using the automatically assigned DNS servers, which you can confirm by visiting [https://1.1.1.1/help](https://1.1.1.1/help) in your browser:

![Debug Information](image)
Section 4. Configuring an Android Phone to Encrypt Stub-to-Recursive-Resolver Traffic

- The latest version of the Android operating system, Android 9 Pie, has out-of-the-box support for DNS over TLS. However, only very limited devices ship with Android 9 Pie (or are even updateable to Android 9 Pie).

- Android will not normally allow you to change the recursive resolvers when you are connecting via mobile service providers. You can set unencrypted third party recursive resolvers when you are connecting over a WiFi connection.

- Until recently, the situation was less favorable if you wanted encrypted recursive resolver service on your Android phone. Until mid-November 2018, Android applications that claimed the ability to encrypt your DNS traffic actually required you to root your device. But attempting to root your device would normally void your phone's warranty (and can also potentially "brick" your handset if the rooting process goes badly). M3AAWG does NOT recommend that users attempt to root their handsets.

- Fortunately, Cloudflare recently released their 1.1.1.1 app via the Playstore (it is somewhat confusingly named—the app has the same name as the IP address of their recursive resolver service). We tested it on a Samsung S6 Edge running Android 7.0 Nougat connected over WiFi, and it worked well. The remainder of this section will show the process of installing the 1.1.1.1 app on Android.

Find the 1.1.1.1 app in the Google Play Store, download it, and install it – it is free.

---

2 https://developers.cloudflare.com/1.1.1.1/setting-up-1.1.1.1/android/
3 See one such list at https://android.gadgethacks.com/news/always-updated-list-phones-will-get-android-pie-0186401/
4 https://support.opendns.com/hc/en-us/articles/228009007-Android-Configuration-instructions-for-OpenDNS
5 https://en.wikipedia.org/wiki/Rooting_(Android)
We'll need to install a VPN Profile

We need to install VPN profile to encrypt your DNS queries.

While the 1.1.1.1 app is enabled, other VPN apps on your device will not work.

Connection request

1.1.1.1 wants to set up a VPN connection that allows it to monitor network traffic. Only accept if you trust the source.

irable at the top of your screen when VPN is active.

1.1.1.1

OFF
Your DNS queries are not private

connected
Your DNS queries are private and faster
M3AAWG Recipes for Encrypting DNS Stub-to-Recursive-Resolver Traffic
Section 5. Configuring a Raspberry Pi Running Raspbian Stretch to Run Unbound and DNS over TLS

Design Goal: Deploy a dedicated local recursive resolver forwarding DNS queries using DNS over TLS

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Dedicated Raspberry Pi B+ (a ≈US$35, single-board computer popular with hobbyists and professionals)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Connectivity</td>
<td>Gigabit Ethernet between the new Raspberry Pi and the user’s existing broadband router</td>
</tr>
<tr>
<td>IP Addressing</td>
<td>Static RFC 1918 (192.168.1.250), sitting alongside other downstream clients behind an existing broadband router</td>
</tr>
<tr>
<td>Operating System</td>
<td>Raspbian Stretch</td>
</tr>
<tr>
<td>DNS Recursive Software</td>
<td>Unbound, performing caching, DNSSEC validation, and forwarding over an encrypted channel</td>
</tr>
<tr>
<td>Third Party Upstream Recursive Software (e.g., the forwarding target)</td>
<td>Cloudflare’s 1.1.1.1. and related IPs</td>
</tr>
<tr>
<td>Encryption of Wide-area DNS traffic</td>
<td>DNS over TLS</td>
</tr>
<tr>
<td>Encryption of Local Traffic</td>
<td>Because the only queries going to the box will be from local clients connecting directly via the RFC 1918 switch fabric, local DNS traffic to/from the recursive resolver will not use encryption (no WiFi directly to the Pi).</td>
</tr>
<tr>
<td>Administrative Traffic</td>
<td>SSH v2 from a private address space-connected device, or https Web portal from private address space</td>
</tr>
</tbody>
</table>

What We Bought: The build described in this section was done using a new CanaKit Raspberry Pi 3 B+ Starter Kit (≈US$80). That kit included:

- Raspberry Pi 3 B+
- Case
- 32GB microSD card preloaded with NOOBS
- Power supply
- HDMI cable
- Heatsinks, etc.

To obtain a graphical interface for initial setup, the user needs to supply an HDMI-capable monitor plus a USB keyboard and mouse.
Hardware Specifications: The Raspberry Pi 3 B+ is the latest Raspberry Pi model, as of this writing, and has impressive specifications relative to some older commercial broadband routers. The Raspberry Pi 3 B+ might be overkill for most consumer-class recursive resolver requirements, but lesser alternatives are not much cheaper and we cannot complain about a board that only costs $≈35. The specifications were:

- ARM Cortex-A53\(^7\) 1.4GHz quad core processor
- 1GB of SDRAM
- A preformatted 32GB microSD card for persistent storage
- NOOBS\(^8\) (New Out Of Box Software, including a full version of Raspbian\(^9\)) preloaded on the microSD card
- 4 USB ports
- 10/100/1000 Ethernet interface (that can reportedly deliver $≈300$Mbps throughput).

The parts are shipped unassembled but it only takes a few minutes to put the system together and set it up. Below are some assembly notes.

System Assembly
- **Caution: Static-sensitive parts!** Ensure that you take appropriate static-electricity-control protective measures.

- We purchased our Raspberry Pi from Cana Kits and the tiny microSD card was shipped in a small pink plastic envelope – take care not to accidentally discard it!

- **Do not** insert the microSD card in its holder on the board until after the case is assembled.

- Did you buy a kit with a preformatted microSD card with NOOBS? You **will not** need to use the microSD card adapter included with the kit – it is only there if you need to reflash the microSD card on your PC or Mac.

- When mounting the Raspberry Pi in the case, separate the case into its three parts (top cover, middle, and bottom cover). **Mount the board onto the bottom cover**, sliding it under the lips as shown at [https://www.canakit.com/pi-case](https://www.canakit.com/pi-case), then attach the middle section as shown in the case assembly video. Do **not** attempt to install the board into the middle part of the case and then snap the bottom cover on. That will not work and can damage the case or the board if you try to force it.

- **Apply** the two, small tape-equipped **heat sinks**. The larger one goes on the CPU. The smaller one goes on the Ethernet chip. (The access hole for the Ethernet chip is offset, but if you fiddle with the heat sink a little, you can get it where it needs to go.)

- Our kit included a **separate in-line switch for the power supply**. 1) Plug the in-line switch into the computer; 2) then plug the power supply into the in-line switch; 3) then plug the switch into the wall.

---

\(^{9}\) [https://www.raspbian.org/](https://www.raspbian.org/)
• The top cover for the case **intentionally does not fit flush** (this gap helps provide airflow). You may want to secure it using a couple of rubber bands around the case to ensure that the top cover stays in place. (Just do not block the ventilation.)

• Connect the monitor, keyboard and mouse. Turn on the power switch. The system should then boot. (The red light on the switch means that the device power is on.)

**Network Configuration and Safety**

There are many different discussions available of what you can and should do to secure your system. We have included some basic recommendations here. Do your own due diligence around this important area. (We also assume you can use some type of file editor.)

**Prompts and Instructions**

In the following instructions, a $ prompt means "this is an unprivileged task, you can run it as any user," while a command shown after a # prompt means "administrative task, either login as root or prefix each such command with sudo." (Naturally, do not actually type in the prompt that is shown before commands.)

• If you want to connect remotely via SSH, you must enable the SSH server.\(^\text{10}\)

• **Reminder:** the root user will normally not be allowed to remotely login with a username and password. If you need to do system administration tasks from the command line, create another regular user account using `adduser`, then `ssh` into that user, then use `su` or `sudo`. Alternatively (but a bad idea), set `PermitRootLogin yes` in `/etc/ssh/sshd_config`.

• If you are going to be connecting to the device via Ethernet, see `/etc/dhcpcd.conf` to define a static IP, netmask and gateway address for `eth0`. (Note that the service's name is `dhcpcd`, not just `dhcpd`). Be sure to set that IP as static on your home broadband router too.

• If tempted to define an initial default recursive resolver in `/etc/resolv.conf`, note Raspbian's use of `resolvconf` (see `/etc/resolvconf.conf`).

• You may want to temporarily set `1.1.1.1` or `9.9.9.9` as a recursive resolver in `/etc/resolvconf.conf`. (This will mean temporarily using regular **unencrypted DNS** service from those services; be sure to eventually remove this.)

• Change the password for the root account, the default pi account, and for any other user accounts you have created.

• Update `/etc/sudoers.d/010_pi-nopasswd` from `pi ALL=(ALL) NOPASSWD: ALL` to `pi ALL=(ALL) PASSWD: ALL`

\(^{10}\) [https://www.raspberrypi.org/documentation/remote-access/ssh/](https://www.raspberrypi.org/documentation/remote-access/ssh/)
• sshd may come with default SSH host keys. Just to ensure that you have your own unique SSH server host keys:
  # rm /etc/ssh/ssh_host_* && dpkg-reconfigure openssh-server

• Consider using SSH preshared keys for SSH authentication (instead of password authentication).11

• Update all the software that is currently installed:
  # apt-get update && apt-get upgrade
  (Slow default mirror? Change that in /etc/apt/sources.list [There is a list of mirrors at https://www.raspbian.org/RaspbianMirrors])

• Run ntp to ensure that system times are synchronized.
  # apt-get install ntp
  # systemctl enable ntp

• Install a current SSL/TLS root certificate bundle:
  # apt-get install ca-certificates

• We like to use the latest-available ("bleeding-edge") distro (Debian-based operating system). Jessie is installed by default, but that is now old. To upgrade to Raspbian Stretch (which is a little better):
  # apt-get update && apt-get dist-upgrade
  Be sure to also update your /etc/apt/sources.list to reflect the fact that you are using Stretch.
  To also update your firmware to the latest bleeding-edge version:
  # apt-get update && apt-get install rpi-update
  # rpi-update
  NOTE: Raspbian Stretch will not use predictable interface names ("eth0") unless you edit
  /boot/cmdline.txt and add net.ifnames=0 to the end of the string that is currently in that file.
  However, there are some indications that doing so may make using WiFi interfaces or multiple Ethernet interfaces problematic (see for example https://www.raspberrypi.org/forums/viewtopic.php?t=192729.) Fortunately, we do not need WiFi and do not use multiple Ethernets.

• Remember to reboot your system.

• Recommended: Install a firewall. We are only connecting this system via RFC1918 address space but we still recommend a firewall.
  ufw12 ("Uncomplicated Fire Wall") is a comparatively-easy-to-use firewall targeting novice firewall users. Here are some quick notes:

11 https://www.raspberrypi.org/documentation/configuration/security.md
12 https://help.ubuntu.com/community/UFW
• # apt-get install ufw
• # ufw enable
  (If ufw will not enable and you upgraded your distro to Stretch, did you run rpi-update? Did you also remember to
  reboot after doing so?)
• # ufw logging low
  (ufw sends its log entries to: /var/log/syslog
  to disable logging: ufw logging off)
• # ufw allow from 192.168.1.0/24 to 192.168.1.250 port 22 proto
tcp
  (allow SSH in from LAN)
• # ufw limit ssh/tcp
  (maximum of half a dozen SSH login attempts per IP in the last half minute)
• # ufw allow from 192.168.1.0/24 to 127.0.0.1 port 53
  (allow DNS queries from LAN hosts)
• # ufw allow from 127.0.0.0/8 to 127.0.0.1 port 53
  (queries from localhost are acceptable, too)
• # ufw status numbered
  (see what ufw rules are active)
• If you added a rule you do not want to use, you can remove it with:
  # ufw delete rule_number_to_del

• Optional: Disable unneeded services started by default.¹³

Installing and Configuring Unbound

Installing Unbound: Raspbian Stretch has Unbound version 1.6.0, which was originally released
December 15, 2016. The current stable version is 1.8.1 (October 8, 2018).¹⁴

Actually installing Unbound as a package is simple:
# apt-get install unbound

Unbound config file (/etc/unbound/unbound.conf): Create the required configuration file using
your favorite editor. We assume that the server was given the static IPv4 address 192.168.1.250 and we
also assume that you want to use Cloudflare's DNS over TLS accessible servers at 1.1.1.1 and 1.0.0.1.
(We have also included commented-out Quad 9's server information and IPv6 equivalent IPs for both.)

¹³https://plone.lucidsolutions.co.nz/hardware/raspberry-pi/3/disable-unwanted-raspbian-services
¹⁴Want to see a list of the bugs patched for each upgraded version? See https://nlnetlabs.nl/projects/unbound/download/ (hit "Toggle Older
Versions" to see the bug fixes since 1.6.0).
server:
  interface: 192.168.1.250
  interface: 127.0.0.1
  port: 53
  do-ip4: yes
  do-ip6: no

  access-control: 0.0.0.0/0 deny
  access-control: 127.0.0.0/8 allow
  access-control: 192.168.1.0/24 allow

  log-time-ascii: yes
  qname-minimisation: yes
  rrset-roundrobin: no

  prefetch: yes
  prefetch-key: yes
  so-reuseport: yes

  hide-identity: yes
  hide-version: yes

forward-zone:
  name: ".

  forward-addr: 1.1.1.1@853#cloudflare-dns.com
  forward-addr: 1.0.0.1@853#cloudflare-dns.com
  # forward-addr: 2606:4700:4700::1111@853#cloudflare-dns.com
  # forward-addr: 2606:4700:4700::1001@853#cloudflare-dns.com
  # forward-addr: 9.9.9.9@853#dns.quad9.net
  # forward-addr: 149.112.112.112@853#dns.quad9.net
  # forward-addr: 2620:fe::fe@853#dns.quad9.net
  # forward-addr: 2620:fe::9@853#dns.quad9.net

  forward-ssl-upstream: yes

include: "/etc/unbound/unbound.conf.d/*.conf"

Once you have created and saved the above file, check it for errors with 
# unbound-checkconf

**Getting DNSSEC Ready for Use:**
Get a copy of the trust anchors for doing DNSSEC by running the 
# unbound-anchor command.

As Unbound's authors insist, compare the values in /var/lib/unbound/root.key with what you see 
from [https://data.iana.org/root-anchors/root-anchors.xml](https://data.iana.org/root-anchors/root-anchors.xml)

**Starting and Testing Unbound:**
You are now ready to try starting Unbound by entering:

# unbound
Test Unbound to see if it is answering queries. You may find dig to be a convenient tool for that testing. Install it with:

```
# apt-get update && apt-get install dnsutils
```

You can then try resolving some sites. Be sure to tell dig to make these queries against the local server, either at 127.0.0.1 or the static IP address assigned to the Raspberry PI (192.168.1.250). For example:

**Test #1: Query of a regular site that does not use DNSSEC. (Note: No ad flag is set in the header):**

```sh
# dig www.m3aawg.org @127.0.0.1
```

```
; <<>> DiG 9.10.3-P4-Raspbian <<>> www.m3aawg.org @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12633
;; flags: qr rd ra;
;; QUERY SECTION:
`;www.m3aawg.org. IN A

;; ANSWER SECTION:
;www.m3aawg.org. 3587 IN CNAME m3aawg.org.
m3aawg.org. 287 IN A 34.214.179.220

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Nov 15 04:35:36 UTC 2018
;; MSG SIZE  rcvd: 73
```

**Test #2: Query for a site that does use DNSSEC. The ad flag should and will be set in the header:**

Now to check to see if Unbound is doing DNSSEC. (We are looking for the ad flag to be set in the header):

```sh
# dig internet2.edu +dnssec +multi @127.0.0.1
```

```
; <<>> DiG 9.10.3-P4-Raspbian <<>> internet2.edu +dnssec +multi @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13430
;; flags: qr rd ra ad;
;; QUERY SECTION:
`;internet2.edu. IN A

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;internet2.edu. IN A
```

*M3AAWG Recipes for Encrypting DNS Stub-to-Recursive-Resolver Traffic*
Test #3: Query an intentionally misconfigured site. We should see a SERVFAIL STATUS returned (and we do).
A domain that has an intentionally or accidentally broken DNSSEC cryptographic validation status should return SERVFAIL, e.g.:

```
# dig www.dnssec-failed.org @127.0.0.1
```

```shell
; <<>> DiG 9.10.3-P4-Raspbian <<>> www.dnssec-failed.org @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 562
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.dnssec-failed.org.  IN A

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Nov 15 04:47:46 UTC 2018
;; MSG SIZE  rcvd: 50
```
Telling the local system to use the locally running copy of Unbound for all queries
Edit: /etc/resolvconf.conf so that: name_servers=127.0.0.1 then run: # resolvconf -u

Telling your local broadband router (and the clients it services) to use the Raspberry Pi's resolver
Go to your local broadband router's Web configuration screen (this is often at https://192.168.1.1) and login. Set the nameservers on that router to point at 192.168.1.250 (or whatever static IP you assigned to the Raspberry Pi). Save the configuration change on the broadband router and reboot it (if it does not reboot automatically). You will only need to do this once.

Important Note:
The architecture described in this section is meant for use on a private (RFC1918) network segment. Even there, clients may still encounter other DNS servers that are suggested by the ISP's DHCP server or seen and remembered by the user's existing broadband router. These other DNS servers may be configured alongside the encrypted recursive resolver described in this section.

There are four main risks associated with this:

- Clients that were meant to use the new encrypted DNS service may end up configured via DHCP to use unencrypted, ISP-provided recursive resolvers. Users may be able to override this by manually configuring just the encrypted local server (e.g., 192.168.1.250) for DNS.

  If the user fails to do this, DNS traffic that was meant to be forwarded over an encrypted connection may actually end up being forwarded unencrypted.

- If users do manually hard code your local recursive resolver's IP, thereby insisting that 192.168.1.250 be used, the connection will not be encrypted. However, this should not be a problem while they are connected to the home network where that device is installed since the Raspberry Pi and the router are directly connected via a private Ethernet cable. Offsite, however, recognize that your RFC1918-only encrypted DNS forwarder will not be available.

  If users are trying to access the RFC1918-only encrypted DNS forwarder, three negative outcomes may occur:

    - Someone else may be running a third party recursive resolver using that same IP address; they may – or may not – be trustworthy.

    - If the user is manually configured to only access a resolver on one specific IP address and there is no recursive resolver accessible on the IP, that user may not be able to resolve any domains until they remove the hardcoded IP address or replace it with an address that is accessible.

    - To prevent user support issues associated with incorrect manually-configured DNS, some broadband routers may redirect all unencrypted DNS traffic to an alternative DNS server of their choice.

Bottom line? Be extremely careful about how you configure your computer's recursive resolvers!
Section 6. Commercial Home Routers Supporting Encryption of Stub-to-Recursive-Resolver Traffic

Most commercial home broadband routers do allow you to specify unencrypted third party recursive resolvers out of the box. Most do not support encrypted third party recursive resolver services out of the box.

One exception is GL.iNet's AR750S\textsuperscript{15} (≈US$70 at the time this was written), which comes pre-configured to do so:

\begin{center}
\begin{tabular}{|l|}
\hline
25+ Commercial VPN Provider Compatibility  \\
Got a commercial VPN provider that you really like? No problem, there's more than 25+ VPNs compatible with our pre-installed OpenVPN client. OpenVPN is a tried-and-true software, and a common standard for security-minded VPN users. WireGuard is based on a concept called Cryptokey Routing. This is an impressive new technology that's worth trying if you're looking for performance boosts beyond what OpenVPN can provide. It's a much faster security protocol than OpenVPN. WireGuard VPN can clock 68Mbps, compared to OpenVPN that can only hit 17Mbps.  \\
Connects Multiple Workplaces Seamlessly  \\
Slate comes with pre-installed VPN server and client functions in WireGuard and OpenVPN. With Slate, you can bring all your workplaces together with a single, secure private cloud with OpenVPN or WireGuard. Don't worry about changing your network when you change your workplace. You won't need to reconfigure and reconnect all your devices as well; they'll all connect automatically each time you fire up your Slate.  \\
Pre-Configured Cloudflare DNS Over TLS  \\
This option uses the TLS security protocol for encrypting DNS queries, which helps increase privacy and prevent eavesdropping. When it comes to browsing privacy, DNS is often the weak link in the chain. Having Cloudflare DNS over TLS is a new feature, previously only possible with professional modification of your router. The Slate comes with this feature baked in; it's as easy as flipping the toggle in the user interface.  \\
\hline
\end{tabular}
\end{center}

\textsuperscript{15}https://smile.amazon.com/GL-iNet-GL-AR750S-Ext-pre-Installed-Cloudflare-Included/dp/B07GBXMBQF/
Section 7. Diagnosing and Testing DNS Stub-to-Recursive Resolver Traffic

**Important Note:** Do NOT run dnstop, tcpdump, Wireshark or any other promiscuous-mode traffic-inspecting tools on M3AAWG-provided connectivity (such as the M3AAWG conference networks). Doing so is explicitly forbidden by the M3AAWG terms of attendance and network acceptable use policies. These tools are ONLY mentioned here for your professional use in validating a resolver installed on a private home network or in conjunction with a resolver on an institutional network that you are authorized to operate and monitor.

1) Check top domain names that are being resolved with dnstop\(^{16}\) by watching live network traffic

Installation of dnstop is often possible via whatever package manager you are normally using on a given platform. For example:

```
# apt-get update && apt-get install dnstop  <-- Raspbian

$ brew install dnstop            <-- Brew on Mac OS X
```

- Once installed, try:
  ```
  # dnstop -l 4 eth0
  ```

- To see the Source IP for the DNS query plus the domain being resolved (for up to four level names) hit $\text{Ctrl}-\text{C}\

$\text{Ctrl}-\text{C}$ interrupts the display. See $\text{man}$ dnstop for more options.

2) Inspect DNS traffic with tcpdump\(^{17}\)

Ensure that you have tcpdump installed. You should be able to install tcpdump using whatever package manager you normally use:

```
# apt-get update && apt-get install tcpdump   <-- Raspbian

$ brew install tcpdump
```

- To see if there is any traffic on port 53…
  ```
  # tcpdump -nt -i eth0 port 53
  ```

Decoding those arguments (see $\text{man}$ tcpdump):

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n</td>
<td>&quot;Do not convert addresses (i.e., host addresses, port numbers, etc.) to names.&quot;</td>
</tr>
<tr>
<td>-t</td>
<td>&quot;Do not print a timestamp on each dump line.&quot;</td>
</tr>
<tr>
<td>-i</td>
<td>&quot;Listen on interface...&quot;</td>
</tr>
<tr>
<td>port 53</td>
<td>&quot;Select just traffic using port 53&quot;</td>
</tr>
</tbody>
</table>

What if you are testing an installation that is sending unencrypted local traffic from local hosts to a local resolver and you see traffic from addresses in 192.168.1.0/24 to addresses 192.168.1.0/24? That is normally acceptable (e.g., assuming the traffic is intra-device, over hard Ethernet links or over encrypted links).

---


\(^{17}\) [http://www.tcpdump.org/](http://www.tcpdump.org/)
• # tcpdump -nt -i eth0 port 853

If you see traffic from addresses in 192.168.1.0/24 to the IP addresses of your upstream resolvers (e.g., 1.1.1.1 or 1.0.0.1) on port 853, or traffic from your specified upstream resolvers (e.g., 1.1.1.1 or 1.0.0.1) to 192.168.1.0/24 on port 853, that is also acceptable and as expected.

• Curious what else is on the network besides port 53 (normal DNS), port 853 (DNS over TLS), or port 22 (ssh)?

# tcpdump -nt -i eth0 port not 53 and port not 853 and port not 22

You will probably see ARP,18 STP,19 UPnP/SSDP traffic on port 1900,20 and ND traffic (IPv6 router advertisements)21. (Now you know some of the reasons why the lights blink even when no one is doing anything.)

Want to focus on what is left? Hide some more of that traffic with:
# tcpdump -nt -i eth0 port not 53 and port not 853 and port not 22 and not arp and not stp and port not 1900

• Coming back to the DNS traffic, want to see still more detail?
# tcpdump -vv -x -X -s 1500 -i eth0 port 53

Decoding the new options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-vv</td>
<td>&quot;Even more verbose output.&quot;</td>
</tr>
<tr>
<td>-x</td>
<td>&quot;When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex. The smaller of the entire packet or snaplen bytes will be printed [...].&quot;</td>
</tr>
<tr>
<td>-X</td>
<td>&quot;When parsing and printing, in addition to printing the headers of each packet, print the data of each packet (minus its link level header) in hex and ASCII.&quot;</td>
</tr>
<tr>
<td>-s 1500</td>
<td>&quot;Snarf snaplen bytes of data from each packet [...].&quot;</td>
</tr>
</tbody>
</table>

3) Verifying/troubleshooting SSL/TLS cert issues

Normally, the major alternative recursive resolver providers will have non-problematic SSL/TLS certificate installations that are readily accessible from most end-user networks or systems. However:

• If you are working from an environment that may be blocking, intercepting, or hijacking DNS-related SSL/TLS traffic, you may need to try to figure out what is going on with that.

• It is also possible that a smaller third party alternative encrypted DNS service might have SSL/TLS certificate issues of one sort or another, such as expired certificates.

• You might be curious if your third-party alternative encrypted DNS service provider's SSL/TLS installation is following best practices, or what sort of certificate they are using.

18 https://en.wikipedia.org/wiki/Address_Resolution_Protocol
Nykolas Z has a Medium.com blog post\textsuperscript{22} that does a good job of introducing the use of the openssl\_client tool and the DNS over TLS-php-client for troubleshooting in a DNS over TLS context.

**Conclusion**

As mentioned, this document provides specific instructions for implementing at least one encrypted recursive resolver solution for each popular platform. For basic information to evaluate the benefits and potential issues with encrypting DNS traffic, see the accompanying document, “M3AAWG Tutorial on Third Party Recursive Resolvers and Encrypting DNS Stub Resolver-to-Recursive Resolver Traffic.”

\textsuperscript{22}https://medium.com/@nykolas.z/troubleshooting-dns-over-tls-e7ea570b6337