2022

Block DNS over HTTPS (DoH), using pfsense

1. About this manual.

If you are reading this document, using Adobe Reader, you may click on a hyperlink to content in this document. Use the combination <Alt> <left arrow> to return to the previous location.

"Back" and "Forward" buttons can also be added to the toolbar. If you right-click on the tool bar, under "Page Navigation", they are referred to as "Previous View" and "Next View".

This document is hosted on GitHub, you can open the document (pdf), using this link.

2. DoH lists.

DNS over HTTPS (DoH) is a protocol for performing remote Domain Name System (DNS) resolution via the HTTPS protocol. You can find a lot of detail on wikipedia.

This document describes a method to prevent (block) clients on your network to use DoH.

In short, we will simply block all the IP's of DoH DNS servers on the firewall. Since DoH servers come and go all the time, it is hard to keep track of all DoH servers. Several contributers dedicate time to creating lists with known DoH servers, unfortunately, most of these lists are incomplete or not maintained.

I've searched and found several lists, containing references to DoH servers:

- https://raw.githubusercontent.com/bambenek/block-doh/master/doh-hosts.txt
- https://raw.githubusercontent.com/Sekhan/TheGreatWall/master/TheGreatWall.txt
- https://raw.githubusercontent.com/oneoffdallas/dohservers/master/list.txt
- https://raw.githubusercontent.com/vysecurity/DoH-Servers/master/README.md
- https://raw.githubusercontent.com/tjay/DoH-List/master/hosts
- https://raw.githubusercontent.com/flo-wer/doh-list/master/domains.txt
- https://raw.githubusercontent.com/wiki/curl/curl/DNS-over-HTTPS.md
- https://download.dnscrypt.info/dnscrypt-resolvers/json/public-resolvers.json
- https://dtm.uk/dns-over-https-doh-servers
- https://raw.githubusercontent.com/Jigsaw-Code/Intra/master/Android/app/src/main/res/values/servers.xml
- https://raw.githubusercontent.com/dibdot/DoH-IP-blocklists/master/doh-ipv4.txt
- https://raw.githubusercontent.com/dibdot/DoH-IP-blocklists/master/doh-ipv6.txt
- https://raw.githubusercontent.com/crypt0rr/public-doh-servers/main/dns.list
- https://raw.githubusercontent.com/xorguy/block-dot-doh/main/Lists/SDNS-Domains.list
- https://raw.githubusercontent.com/jbaggs/doh-intel/master/doh.intel
- https://raw.githubusercontent.com/mili-tan/ArashiDNS.Dekunua/main/DoH.list

Feel free to report any other list you find.

The above lists are used to build new lists, containing IP adresses, that can be used on a firewall. This process runs daily, the resulting github files will be updated, as soon as a change is detected.

This document describes what you need to do to use the IP lists on pfsense. If your firewall doesn't support this, you might want to add the DoH response policy zone (unbound, bind, knot resolver, …).

3. **Pfsense configuration.**

There are different ways to use the IP block lists and IP exceptions lists. Any solution will do, as long as the result is DoH being blocked, except for the specific devices in the exception rules.

I use floating rules for all LAN interfaces, except the interface I'm using in my test environment. I don't want to define additional rules on the WAN interface, to allow access to all DoH IPs on the test interface, hence targeting the LAN interface. This document describes the approach for a single LAN interface.

1. Defining the firewall aliases URLs.
   The Github repository contains four relevant files, we need to create URL aliases for each list. The lists (how to use them will be explained later):

   - DOHipv4.txt: This list contains the IPv4 addresses of al DoH servers found in the lists.
   - DOHexceptionsIPv4.txt: This list contains the IPv4 addresses of DoH servers, that also provide a service or content on the same Ipv4 address.
   - DOHipv6.txt: This list contains the IPv6 addresses of al DoH servers found in the lists.
   - DOHexceptionsIPv6.txt: This list contains the Ipv6 addresses of DoH servers, that also provide a service or content on the same Ipv6 address.

Goto 'Firewall / Aliases / URL's' and click 'Add'

Create all (four) the aliases (two if you're not using IPv6), using the data from the table, ensure type 'URL Table (IPs)' is selected, set the update frequency to '1'. Feel free to choose a name, description, the document will use the names from the table.

Name	URL
DoHserversIPv4	https://raw.githubusercontent.com/jpgpi250/piholemanual/master/DOHipv4.txt
DoHserversIPv6	https://raw.githubusercontent.com/jpgpi250/piholemanual/master/DOHipv6.txt
DoHserverExceptionsIPv4	https://raw.githubusercontent.com/jpgpi250/piholemanual/master/DOHexceptionsIPv4.txt
DoHserverExceptionsIPv6	https://raw.githubusercontent.com/jpgpi250/piholemanual/master/DOHexceptionsIPv6.txt

The result should look like this (example):

2. Defining the DoH block rules.

Goto 'Firewall / Rules / WAN' and click 'Add (rule to the top of the list)

- Action: Block

- Select Quick

- Interface: LAN

- Address Family: Select the correct Address family (list IPv4 or IPv6)

- Protocol: TCP/UDP

Protocol	TCP/UDP ▼

- Source: Any

Source	☐ Invert match	any ▼

- Destination:
  Single host or alias, select the correct alias (DoHserversIPv4 or DoHserversIPv4)

Destination	☐ Invert match	Single host or alias ⌄	DoHserversIPv4

- Destination Port Range: 443

Destination Port Range	HTTPS (443) ▼		HTTPS (443) ▼
	From	Custom	To

- Optionally, enable logging and enter a comment.

You should now have 2 rules (if you're using IPv6), the result should look like this (I have logging enabled, hence the extra icon):

☐	States	Interfaces	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
☐ ✖ ⏩ ≣	0 /571 KiB	LAN	IPv4 TCP/UDP	*	*	DoHserversIPv4	443 (HTTPS)	*	none		block DoHipv4 port 443
☐ ✖ ⏩ ≣	0 /659 KiB	LAN	IPv6 TCP/UDP	*	*	DoHserversIPv6	443 (HTTPS)	*	none		block DoHipv6 port 443

If you have a firewall with multiple LAN ports, you could choose to use floating rules, allowing a specific adapter to use DoH (for test purposes).

A floating rule has an additional field.

- Direction: any

Direction	any ▼

A floating rule configuration would look like this (the WAN and OPT2 adapter have been excluded from the rule, this allows me to test DoH on a specific network segment):

| ☐	States	Interfaces	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
☐ ✖ ⏩ ≣	0 /571 KiB	LAN OPT1 WIFI	IPv4 TCP/UDP	*	*	DoHserversIPv4	443 (HTTPS)	*	none		block DoHipv4 port 443
☐ ✖ ⏩ ≣	0 /659 KiB	LAN OPT1 WIFI	IPv6 TCP/UDP	*	*	DoHserversIPv6	443 (HTTPS)	*	none		block DoHipv6 port 443

3. Defining the exception alias

The entries on the list are now blocked, using the above rules. Unfortunately the IP for the domain dns.cloudflare.com (example) is the same as the IP for cdnjs.cloudflare.com. This will break the browsing experiance. We need to ensure browsers can load pages that use content from this IP, whithout simply removing the entry from the list.

Examples of web pages that require the exception:

- rtvnoord.nl
- linuxquestions.org
- liveleak.com

Looking at, for example rtvnoord.nl, it's clear the site uses content from cdnjs.cloudflare.com, which is behind the same IP as dns.cloudflare.com.
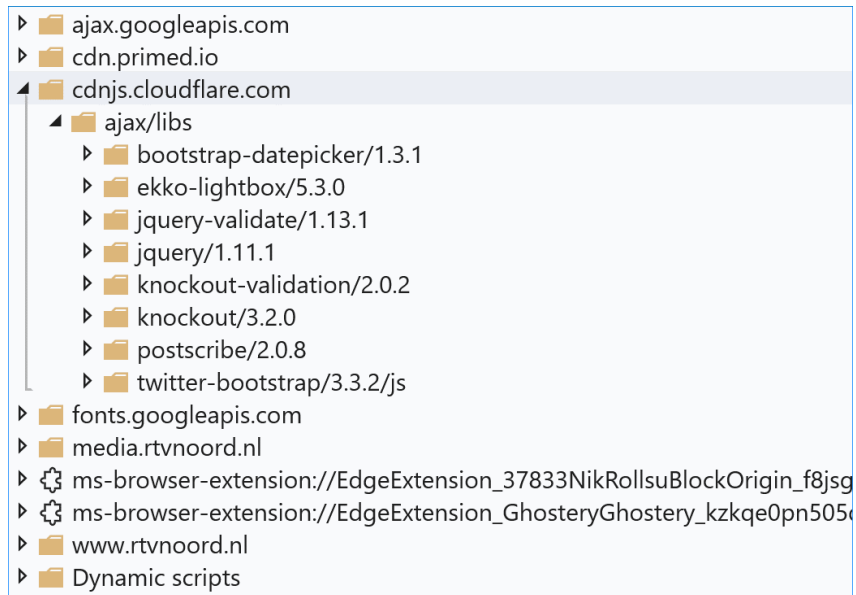
After defining the above block rules, browsing to www.rtvnoord.nl (a dutch web site), the following firewall entries will show up in the log (if logging is enabled).

The IP addresses for dns.cloudflare.com are included in DOHexceptionsIPv4.txt and DOHexceptionsIPv6.txt. We already created aliases for these lists (see defining the firewall aliases URLs).

The idea is to use these lists (exceptions) to allow **specific devices** to bypass the DoH block rules, defined in the previous section. This disables DoH protection, using the firewall rules, for the specific devices only.

As an alternative, if you do NOT want to create exception aliases, you can always download the IP lists on the local machine, and remove the entries, using a script (wget or curl to download, sed to remove entries).

In small environments, the easiest way to maintain an alias with the IPs of specific devices, is to create an IP alias:

Goto 'Firewall / Aliases / IP' and click 'Add'

Create all (two) the aliases, using the data from the table, ensure type 'Host(s)' is selected.
Feel free to choose a name, description, the document will use the names from the table.

Name	Description
DoHclientExceptionsIPv4	IPv4 hosts, allowed to bypass specific DoH IPv4 entries
DoHclientExceptionsIPv6	IPv6 hosts, allowed to bypass specific DoH IPv6 entries

Add IP addresses and descriptions as needed.

The result should look like this (example, use your own IP addresses or subnet):

4. Defining the exception rules.

Create the firewall rules to allow specific devices to connect to specific IP's, port 443 (example dns.cloudflare.com, see above).

Goto 'Firewall / Rules / WAN' and click 'Add (rule to the top of the list)

- Action: Pass

- Select Quick

- Interface: WAN

- Address Family: Select the correct Address family (list IPv4 or IPv6)

Address Family

IPv4

- Protocol: TCP/UDP

Protocol

TCP/UDP

- Source:
Single host or alias, select the correct alias (DoHclientExceptionsIPv4 / DoHclientExceptionsIPv6)

Source

Source	☐ Invert match	Single host or alias ∨	DoHclientExceptionsIPv4

- Destination:
Single host or alias, select the correct alias (DoHserverExceptionsIPv4 / DoHserverExceptionsIPv6)

Destination

Destination	☐ Invert match	Single host or alias ∨	DoHserverExceptionsIPv4

- Destination Port Range: 443

Destination Port Range	HTTPS (443) ▾		HTTPS (443) ▾
	From	Custom	To

- Optionally, enable logging and enter a comment.

You should now have 4 rules, 2 allow rules, 2 block rules (if you're using IPv6). You can change the order of the rules, by dragging the rule to the desired position (don't forget to save, after changing the order). The result should look like this:

Rules (Drag to Change Order)

☐	States	Interfaces	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
☐ ✔ ⏩	0 /8.63 MiB	LAN	IPv4 TCP/UDP	DoHclientExceptionsIPv4	*	DoHserverExceptionsIPv4	443 (HTTPS)	*	none		allow DoHipv4 port 443
☐ ✖ ⏩	0 /571 KiB	LAN	IPv4 TCP/UDP	*	*	DoHserversIPv4	443 (HTTPS)	*	none		block DoHipv4 port 443
☐ ✔ ⏩	0 /0 B	LAN	IPv6 TCP/UDP	DoHclientExceptionsIPv6	*	DoHserverExceptionsIPv6	443 (HTTPS)	*	none		allow DoHipv6 port 443
☐ ✖ ⏩	0 /659 KiB	LAN	IPv6 TCP/UDP	*	*	DoHserversIPv6	443 (HTTPS)	*	none		block DoHipv6 port 443

Again, if you have a firewall with multiple LAN ports, you could choose to use floating rules.

4. **List Updates.**

The lists are generated and pushed to GitHub (if there are changes) daily at 04:30 UTC, pfsense refreshes the content of URL Table IPs aliases, using a cron job.

I have modified this cron job, this to ensure pfsense will be using the updated lists at the beginning of the workday.

44	7	*	*	*	root	/usr/bin/nice -n20 /etc/rc.update_urltables now forceupdate

Optionally, install the cron package (System / Package Manager) to change the cron job.

It's also possible to create a script to force update the lists.

Create a script /root/updatelists.sh and make it executable, content:

```
/usr/bin/nice -n20 /etc/rc.update_urltables now forceupdate
```

Selecting option 8 (Shell) on the pfsense console will allow you to execute:

```
./updatelists.sh
```

You can verify the successful execution, using the web interface (Status / System Logs / System / General), you will see something like this (using local copy of the lists).

Jun 16 10:06:47	php-cgi	rc.update_urltables: /etc/rc.update_urltables: Updated DOHipv6 content from http://192.168.2.57/DOHservers/DOHipv6.txt: no changes.
Jun 16 10:06:47	php-cgi	rc.update_urltables: /etc/rc.update_urltables: Updated DOHipv4 content from http://192.168.2.57/DOHservers/DOHipv4.txt: no changes.

5. **Sqlite3 database.**

   The lists are generated, using dig replies in my region (Lat.Long 51.2211097,4.3997081). Unfortunately, some domains (example dns.nextdns.io) resolve into a region dependent IP address. The IP(s) for nextdns.io, retrieved in my region are thus blocked, but these addresses may not block DOH in your region. To overcome this problem, and allow users to generate localized DOH block lists, I have created a database, containing the information, used to generate the IP lists.

   You might want to read section 6 (Network control, using DNS entries). It also contains information regarding localized replies for the domains *mask.apple-dns.net* and *mask-t.appledns.net.* The files in the GitHub repository (DOHipv4.txt and DOHipv6.txt only contain relay entries for my region (Lat.Long 51.2211097,4.3997081).

   The database can be accessed, using *sqlite3* commands (OR *pihole-FTL sqlite3*), to make life easier, take a look at my pi-hole manual, section 26/3 (Browsing the FTL database). You'll need to make some changes in */var/www/html/phpliteadmin.config.php*, this to include the new database. A script to install this on Raspberry Pi OS Lite can be found here.

   1. Tables
      - **info**:
        o version: If you're going to write a script to build your own lists, ensure to verify the database version. The version may change, as I'm still trying to improve the quality of the resulting lists.
        o latest_timestamp: used to identify entries that are added or updated during the last run (daily).

- **urllist**: contains the url of the source lists, see DOH lists. Every list has an id, used to identify the list. The timestamp indicates when the last change to the list was detected. A 'null' value indicates there has been no change since this field was added to the database (version 2). The timestamp of new lists is also null.
- **domainlist**: contains all of the domains found in all of the lists, which implies there are duplicates. Example: the domain "*dns.adguard.com*" is found in 11 lists, thus has 11 domainlists entries. See query examples to retrieve this info.

```
1  https://raw.githubusercontent.com/Sekhan/TheGreatWall/master/TheGreatWall.txt|dns.adguard.com
2  https://raw.githubusercontent.com/oneoffdallas/dohservers/master/list.txt|dns.adguard.com
3  https://raw.githubusercontent.com/flo-wer/doh-list/master/domains.txt|dns.adguard.com
4  https://raw.githubusercontent.com/wiki/curl/curl/DNS-over-HTTPS.md|dns.adguard.com
5  https://download.dnscrypt.info/dnscrypt-resolvers/json/public-resolvers.json|dns.adguard.com
6  https://raw.githubusercontent.com/dibdot/DoH-IP-blocklists/master/doh-ipv4.txt|dns.adguard.com
7  https://raw.githubusercontent.com/dibdot/DoH-IP-blocklists/master/doh-ipv6.txt|dns.adguard.com
```

However, the unique_id will be equal for all entries.

Every entry also has a timestamp, which is updated, whenever the lists are processed (daily). If a domain is no longer on a source list, the timestamp will NOT be updated.

Several methods are used to extract the domains from the source lists (different format such as plain text, html, json, …). As a result, some domains need to be excluded from the list. Example: "*github.com*" is mentioned on several html pages, this domain will thus be excluded (not added to the database). Another example: "*meganerd.nl*" is a valid domain, the DOH server ("*chewbacca.meganerd.nl*", according to the lists) has a different IP address. Thus, "*meganerd.nl*" is excluded (not added to the database), "*chewbacca.meganerd.nl*" is included.
- **exceptions**: domains that will need an exception (allowed rule) for some devices, in order to allow some websites to load without delay (some websites will not load at all without this exceptions).
  o domain: The domain you need to add an exception for
  o timestamp: used to identify entries that are added or updated during the last run (daily).
- **cnameinfo**: every unique domain entry is processed (dig), this to retrieve IP addresses. Some of the list entries are actually CNAMEs. Example: the lists contain an entry "*adblock.lux1.dns.nixnet.xyz*". This is actually a cname for "*lux1.nixnet.xyz*". This information is stored in the cnameinfo table. Be aware the cnameinfo table does NOT contain the CNAME(s), only the relevant info, as a result from the DNS query.

The domainlist_id points to the entry in the domainlist table.

The timestamp is updated every time the lists are processed (daily). If a DNS query no longer returns this info, the timestamp will NOT be updated.

2. Query examples

I'm NOT a database expert, so there may be room for improvement… Some examples, I use, to get relevant information from the database. The examples are formatted to be used in a bash script (#!/bin/bash).

You need to change the location of the database to reflect the actual location.

- Get last detected list change date ('null' implies no change since 29-03-2021)

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT address, date(timestamp, 'unixepoch') FROM urllist;"
```

- Get list(s) that contain specific domain:

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT address, domain FROM 'domainlist' \
    INNER JOIN 'urllist' ON urllist.id = domainlist.urllist_id \
    WHERE domainlist.domain = 'dns.adguard.com';"
```

- Get domains for a specific CNAMEinfo entry

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT domain, urllist_id, cname_domain, unique_id FROM 'cnameinfo' \
    JOIN 'domainlist' ON domainlist_id = domainlist.unique_id \
    WHERE cnameinfo.cname_domain = 'lux1.nixnet.xyz.';"
```

- Get unique domains

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT DISTINCT domain FROM 'domainlist';"
```

OR, using the 'latest_timestamp' value (only list the domains detected in the last version of the source lists):

```
sudo sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT DISTINCT domain FROM 'domainlist' \
    WHERE domainlist.timestamp = \
        (SELECT value FROM info \
        WHERE property = 'latest_timestamp');"
```

This query result can be used to generate a custom pi-hole list. Redirect the output to a file, a new local pi-hole blocklist, thus:

```
sudo sqlite3 /home/pi/DOH/sqlite3/DOH.db \
    "SELECT DISTINCT domain FROM 'domainlist' \
    WHERE domainlist.timestamp = \
        (SELECT value FROM info \
        WHERE property = 'latest_timestamp');" \
    | sudo tee /var/www/html /DOHdomains.txt
```

Add a local pi-hole list (Group Management / Adlists), do NOT include the quotes:
  - *"http://localhost/DOHdomains.txt"*
  OR
  - *"file:///var/www/html/DOHdomains.txt"*

Pi-hole users can check if any of their clients has submitted a DNS query to pi-hole by executing the following query, **you don't have to use the proposed blocklist to run this query**:

```
sqlite3 "/home/pi/DOH/sqlite3/DOH.db" \
        "ATTACH database '/etc/pihole/pihole-FTL.db' as 'FTLdb'; \
        SELECT DISTINCT domainlist.domain, queries.client \
        FROM domainlist \
        JOIN FTLdb.queries ON domainlist.domain = queries.domain;"
```

Ideally, this would return an empty result. I've been testing a lot on both my pi and workstation, my (partial) result looks like this:

```
adblock.lux1.dns.nixnet.xyz|127.0.0.1
adblock.lux1.dns.nixnet.xyz|192.168.2.228
chewbacca.meganerd.nl|127.0.0.1
chewbacca.meganerd.nl|192.168.2.227
doh.captnemo.in|127.0.0.1
doh.captnemo.in|192.168.2.228
lux1.nixnet.xyz|127.0.0.1
```

- Get unique domains, display all fields

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
        "SELECT * FROM 'domainlist' group by domain;"
```

- Get unique domains, display all fields, order by specific field

```
sqlite3 /home/pi/DOH/sqlite3/DOH.db \
"SELECT * FROM 'domainlist' \
        GROUP BY domain \
        ORDER BY unique_id;"
```

6. **Network control, using DNS entries.**
   Unfortunately, some lists already contain entries for domains that should NOT be blocked. These domains are used for controlling the use of DoH on a network level. If the DNS queries are not answered correctly (NXDOMAIN), blocking them will cause delays.

   1. Firefox canary domain.

      Firefox has provided a method to prevent the browser from using DoH.

      In order to ensure the browser doesn't use DoH, you should ensure the DNS query for the domain *use-application-dns.net* is answered with NXDOMAIN.

      In the latest pi-hole version, the reply for this domain is hardcoded, so no additional action is needed.

      If you don't use pi-hole and the DNS reply for this domain (dig use-application-dns.net) isn't NXDOMAIN, you can:

- Use dnsmasq to ensure the query is answered correctly, by adding a dnsmasq configuration line:

```
server=/use-application-dns.net/
```

- Use unbound to ensure the query is answered correctly, by adding unbound configuration lines:

```
server:
    local-zone: " use-application-dns.net." always_nxdomain
```

2. iCloud Private Relay.

Apple uses an implementation of oDOH (oblivious DoH), this to increase privacy. When enabled (default), the system no longer uses the configured DNS server(s), provided by DHCP. Apple has provided a method to ensure users get a warning. The user than needs to disable the use of iCloud Private Network for the network he is connected to.

Example, warning for WLAN (NXDOMAIN reply for the domain(s) received), the user needs to select "Use without Private Relay", this to avoid using oDoH (icloud Private Relay).
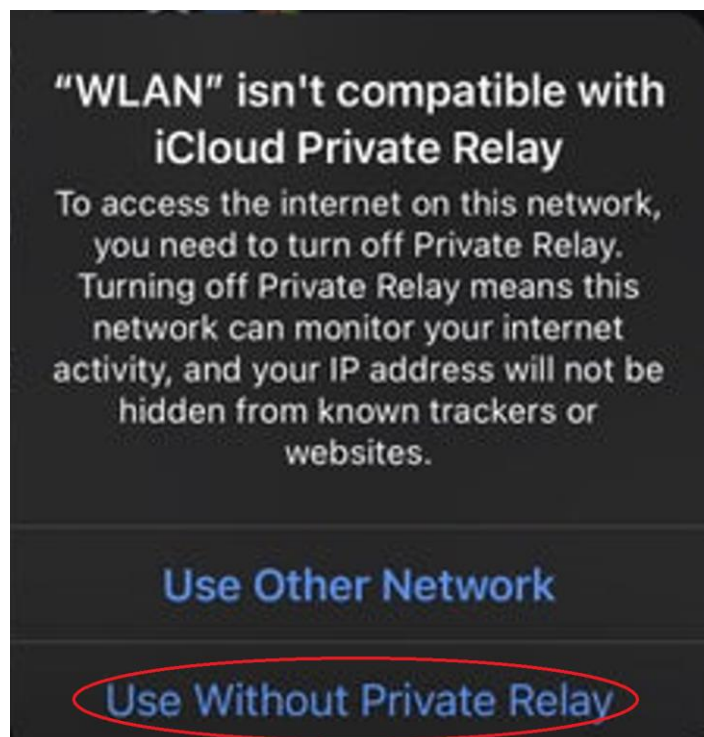
In order to ensure the system does warn the user about iCloud Private Relay(s) (oDoH), you should ensure the DNS queries for the domains *mask.apple-dns.net* and *mask-t.appledns.net* are answered with NXDOMAIN.

In the latest pi-hole version, the reply for these domains is hardcoded, so no additional action is needed.

If you don't use pi-hole and the DNS reply for these domains isn't NXDOMAIN, you can:

- Use dnsmasq to ensure the queries are answered correctly, by adding dnsmasq configuration lines:

```
server=/mask.icloud.com/

server=/mask-h2.icloud.com/
```

- Use unbound to ensure the queries are answered correctly, by adding unbound configuration lines:

```
server:

    local-zone: " mask.icloud.com." always_nxdomain

    local-zone: " mask-h2.icloud.com." always_nxdomain
```

The normal response (not yet changed to NXDOMAIN) for the above domains is localized, the response contains the addresses of the local available relays, the system will use, different for each region. If you want to ensure the relays in your region are blocked, using the firewall rules, you need to generate the IP lists yourself, using the database, see section 5 (Sqlite3 database). A list of all relays can be found here.

7. **DoH response policy zone.**
Not all firewalls are capable of implementing this. To provide protection for (o)DoH, the domains (valid IP found) are added to a rpz file, that can be used to ensure unbound (bind and knot resolver also support this – not tested) returns an appropriate response for these domains. More on this in this document.

The latest version of the rpz file contains policies for (o)DoH domains **and** their IP addresses, this to prevent accessing (o)DoH services, using an alternative DNS name.

The unbound documentation for response policy zones can be found here.

8. **Change Log.**
16-06-2020

- Initial release.

11-07-2020

- Added list from Alphabet's Intra app, ref GitHub issue 4.

17-01-2021

- Entries from the existing (previous version) lists are only kept if the IP can be resolved, using the OpenDNS resolvers.

09-02-2021

- Removed list from heuristicsecurity.com (https://*heuristicsecurity.com/dohservers.txt*), this URL is redericted to a non-DoH related page.
- Added Lists from https://*github.com/dibdot/DoH-IP-blocklists*, ref GitHub issue 8.

28-02-2021

- Added the database to the GitHub repository.

01-03-2021

- Added the 'latest_timestamp' property (info table).
- Added query example to use the 'latest_timestamp' value.
- Additional info for pi-hole users.

29-03-2021

- Added 'timestamp' field to 'urllist' table, this to identify unmaintained lists. A 'null' value indicates the list hasn't changed, since this field was added to the table.
- Added query example to show human readable list change date.
- Increased database version (new field).

10-04-2021

- Added 'exceptions 'table, to identify domains that need an exception for some devices.
- Added 'doh1.b-cdn.net' and 'doh2.b-cdn.net' to the exceptions list (blocking the IP address prevents loading of some sites, such as discourse.pi-hole.net.
- Increased database version (3).

20-09-2021

- Added List https://*raw.githubusercontent.com/crypt0rr/public-doh-servers/main/dns.list*.

01-10-2021

- Added section Network control, using DNS entries.

21-12-2021

- Removed **0.0.0.0** and **::** entries (return value for dns.dnsoverhttps.net).
- Added automatic cleanup of database cnameinfo entries.

02-01-2022

- Added section DoH response policy zone.
- Daily updated DOH.rpz on Github.

02-08-2022

- Added IP policies to the response policy zone.

- Added new lists:
    - https://raw.githubusercontent.com/xorguy/block-dot-doh/main/Lists/SDNS-Domains.list
    - https://raw.githubusercontent.com/jbaggs/doh-intel/master/doh.intel
    - https://raw.githubusercontent.com/mili-tan/ArashiDNS.Dekunua/main/DoH.list