DNS over HTTPS

How to build a DNS over HTTPS proxy in PHP

Alberto Di Maio [lbrtdm@gmail.com]

Full Stack Developer

What?

From Wikipedia:

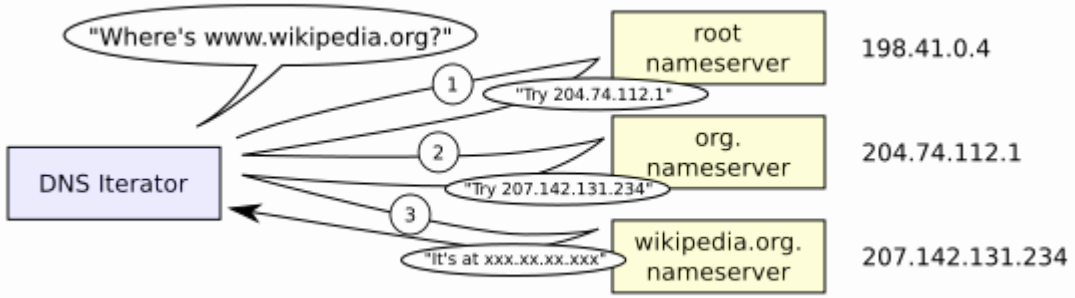
> "DNS over HTTPS (DoH) is an experimental protocol for performing remote Domain Name System (DNS) resolution via the HTTPS protocol. The goal of the method is to increase user privacy and security by preventing eavesdropping and manipulation of DNS data by man-in-the-middle attacks."

DNS

RFCs: https://www.isc.org/community/rfcs/dns/

From Wikipedia:

> "The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System is an essential component of the functionality on the Internet, that has been in use since 1985."



HTTPS

RFCs: HTTP Over TLS TLS 1.2 HTTP 1.1 HTTP 2

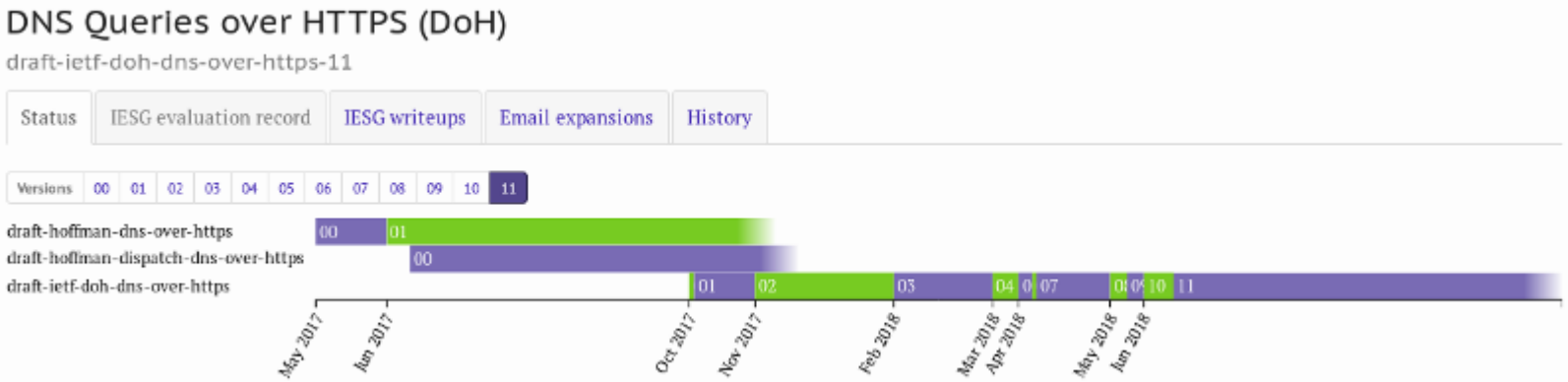
From Wikipedia:

> "HTTP Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP) for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS), or formerly, its predecessor, Secure Sockets Layer (SSL). The protocol is therefore also often referred to as HTTP over TLS, or HTTP over SSL. The principal motivation for HTTPS is authentication of the accessed website and protection of the **privacy** and **integrity** of the exchanged data while in transit. It protects against *man-in-the-middle attacks*. The bidirectional encryption of communications between a client and server protects against eavesdropping and tampering of the communication. In practice, this provides a reasonable assurance that one is communicating without interference by attackers with the website that one intended to communicate with, as opposed to an impostor."

Status: Internet Draft v11

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF).

[HTML] https://datatracker.ietf.org/doc/draft-ietf-doh-dns-over-https/



Requirements

- The protocol must use normal HTTP semantics.
- The protocol must use a secure transport that meets the requirements for HTTPS.
- The protocol must ensure interoperability by specifying a single format for requests and responses that is mandatory to implement.
- HTTP/2 RFC7540 is the minimum RECOMMENDED version of HTTP for use with DoH.

Non requirements

- Supporting insecure HTTP

Request

- URI RFC6570
- HTTP GET or POST
- The URI Template is processed without any variables when the HTTP method is POST.
- When the HTTP method is GET the single variable "dns" is defined as the content of the DNS request (as described in Section 7), encoded with base64url RFC4648.

Request Example

GET:

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query?dns=AAABAAABAAAAAAAAA3d3dwdleGFtcGxlA2NvbQAAAQAB
accept = application/dns-message
```

POST:

```
:method = POST
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/dns-message
content-type = application/dns-message
content-length = 33

<33 bytes represented by the following hex encoding>
00 00 01 00 00 01 00 00  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
01
```

Response

- The only response type defined is "application/dns-message", but it is possible that other response formats will be defined in the future.
- A successful HTTP response with a 2xx status code RFC7231 can be used for any valid DNS response
- HTTP responses with non-successful HTTP status codes do not contain replies to the original DNS question in the HTTP request.
- The data payload for the application/dns-message media type is a single message of the DNS on-the-wire format (originally for DNS over UDP) defined in section 4.2.1 of RFC1035

Response Example

This is an example response for a query for the IN A records for "www.example.com" with recursion turned on. The response bears one record with an address of 192.0.2.1 and a TTL of 128 seconds.

```
:status = 200
content-type = application/dns-message
content-length = 64
cache-control = max-age=128

<64 bytes represented by the following hex encoding>
00 00 81 80 00 01 00 01  00 00 00 00 03 77 77 77
07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
01 03 77 77 77 07 65 78  61 6d 70 6c 65 03 63 6f
6d 00 00 01 00 01 00 00  00 80 00 04 C0 00 02 01
```

How?

Amp

Project website: https://amphp.org/

Amp is a non-blocking concurrency framework for PHP providing primitives to manage concurrency such as an event loop, promises, and asynchronous iterators.

Main components:

- **Event Loop** The event loop is the main task scheduler of every asynchronous application. It dispatches associated handlers once the registered events happen.
- **Promises** A promise is a placeholder for the result of an asynchronous operation. While synchronous programs block until a result is available, asynchronous programs return a placeholder which gets filled in with the result at a later time.
- **Coroutines** Coroutines are interruptible functions. In PHP they can be implemented using generators. While generators are usually used to implement simple iterators and yielding elements using the `yield` keyword, Amp uses `yield` as interruption points. When a coroutine yields a value, execution of the coroutine is temporarily interrupted, allowing other tasks to be run, such as I/O handlers, timers, or other coroutines.

Aerys

Project website: https://amphp.org/aerys/

Aerys is a non-blocking HTTP/1.1 and **HTTP/2** application, WebSocket and static file server written in PHP.

Install:

```
composer require amphp/aerys
```

Run:

```
vendor/bin/aerys -d -c config.php
```

Example:

```php
// config.php

return (new Aerys\Host)
    ->use(function(Aerys\Request $request, Aerys\Response $response) {
        $response->end("<h1>Hello, world!</h1>");
    });
```

Talk is cheap. Show me the code.

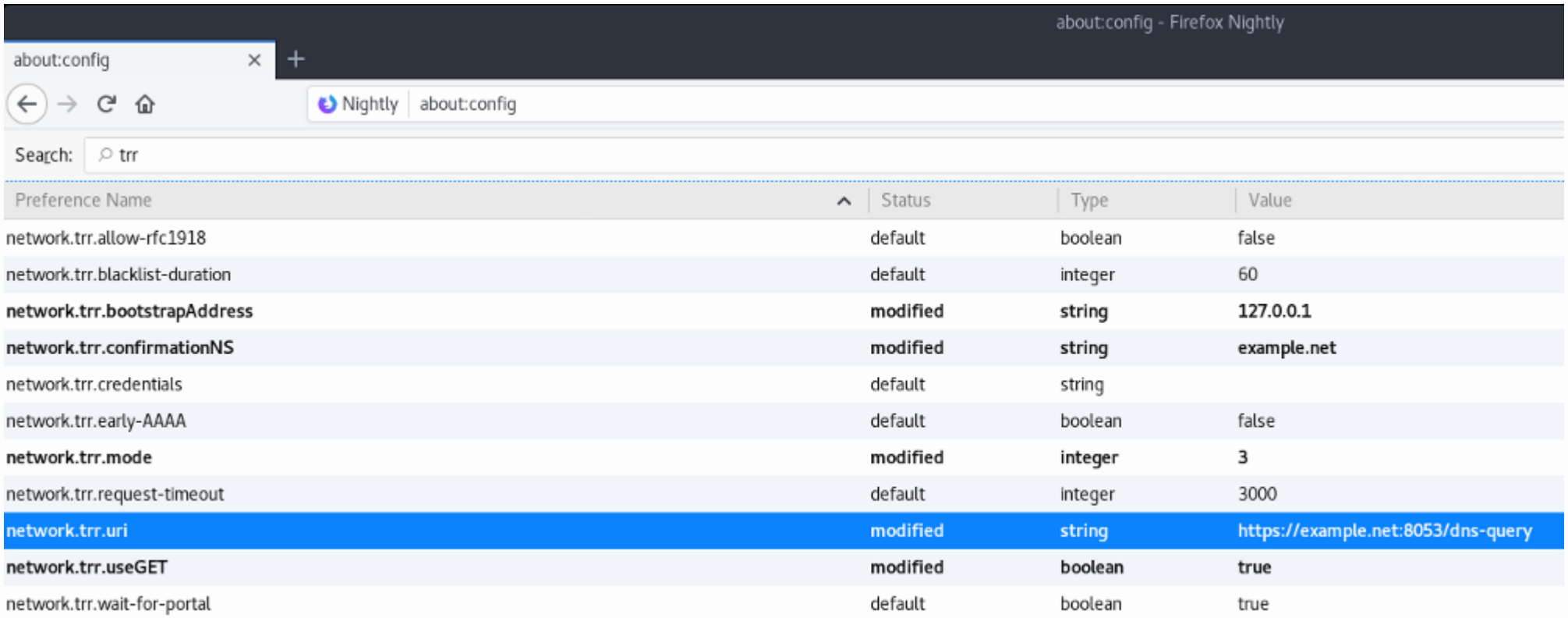
Linus Torvalds on Linux Kernel Mailing List

```php
// config.php

use Aerys\{ Http2Driver, Host, function root, function router };
use \DoH\{Handler, Logger};

const AERYS_OPTIONS = [
    "connectionTimeout" => 60,
    "deflateMinimumLength" => 0,
    "sendServerToken" => true,
];

$router = router()
    ->route("GET", "dns-query", [(new Handler), 'get']);

return (new Host)
    ->use(new Logger)
    ->expose("*", 8053)
    ->name('example.net')
    ->use(new Http2Driver)
    ->encrypt('./certs/fullchain.pem', './certs/privkey.pem')
    ->use($router);
```

```php
// Handler.php

use Aerys\Request;
use Aerys\Response;
use function Amp\Socket\connect;

class Handler
{
    CONST SERVER = '127.0.0.1';
    CONST PORT = 53;

    public function get(Request $request, Response $response)
    {
        $dnsRequest = self::base64UrlDecode($request->getParam('dns'));
        $dnsResolver = \sprintf("udp://%s:%d", self::SERVER, self::PORT);
        $socket = yield connect($dnsResolver);
        yield $socket->write($dnsRequest);
        $responsePacket = yield $socket->read();
        $response->setHeader("Content-type", "application/dns-udpwireformat");
        $response->end($responsePacket);
    }

    public static function base64UrlDecode(string $data)
    {
        return base64_decode(strtr($data, '-_', '+/'));
    }
}
```

```php
// Logger.php

use Aerys\Bootable;
use Aerys\Request;
use Aerys\Response;
use Aerys\Server;
use DoH\Handler;
use LibDNS\Decoder\DecoderFactory;
use Psr\Log\LoggerInterface;

class Logger implements Bootable {

    private $logger;

    function boot(Server $server, LoggerInterface $logger) {
        $this->logger = $logger;
    }

    function __invoke(Request $request, Response $response) {
        $dnsParam = $request->getParam('dns');
        if ($dnsParam) {
            $dnsRequest = Handler::base64UrlDecode($dnsParam);
            $dnsDecoder = (new DecoderFactory)->create();
            $decodedRequest = $dnsDecoder->decode($dnsRequest);
            $questions = $decodedRequest->getQuestionRecords();
            foreach ($questions as $record) {
                $this->logger->info($record->getName());
            }
        }
    }
}
```

Why?


about:config - Firefox Nightly
about:config
Nightly
about:config
Search:
trr
Preference Name
Status
Type
Value
network.trr.allow-rfc1918
default
boolean
false
network.trr.blacklist-duration
default
integer
60
network.trr.bootstrapAddress
modified
string
127.0.0.1
network.trr.confirmationNS
modified
string
example.net
network.trr.credentials
default
string
network.trr.early-AAAA
default
boolean
false
network.trr.mode
modified
integer
3
network.trr.request-timeout
default
integer
3000
network.trr.uri
modified
string
https://example.net:8053/dns-query
network.trr.useGET
modified
boolean
true
network.trr.wait-for-portal
default
boolean
true

Resources

- Improving DNS Privacy in Firefox
- A cartoon intro to DNS over HTTPS
- CloudFlare DNS Resolver
- DNS over HTTPS on curl wiki

Q&A

Bye bye!