Secure WordPress

Step up your game with HTTP headers

LEVEL LEVEL

Bernard Zijlstra
WordCamp Rotterdam 2019

My name is Bernard Zijlstra

My job title in Slack is “Sandbox Master”. Using a more generic and popular term, one would call me the CTO.

Download slides:

lvl.li/htmlheaders

(don’t worry, the link is on every slide)

Find us on:

@levellevel

So, what is this HTTP thing?

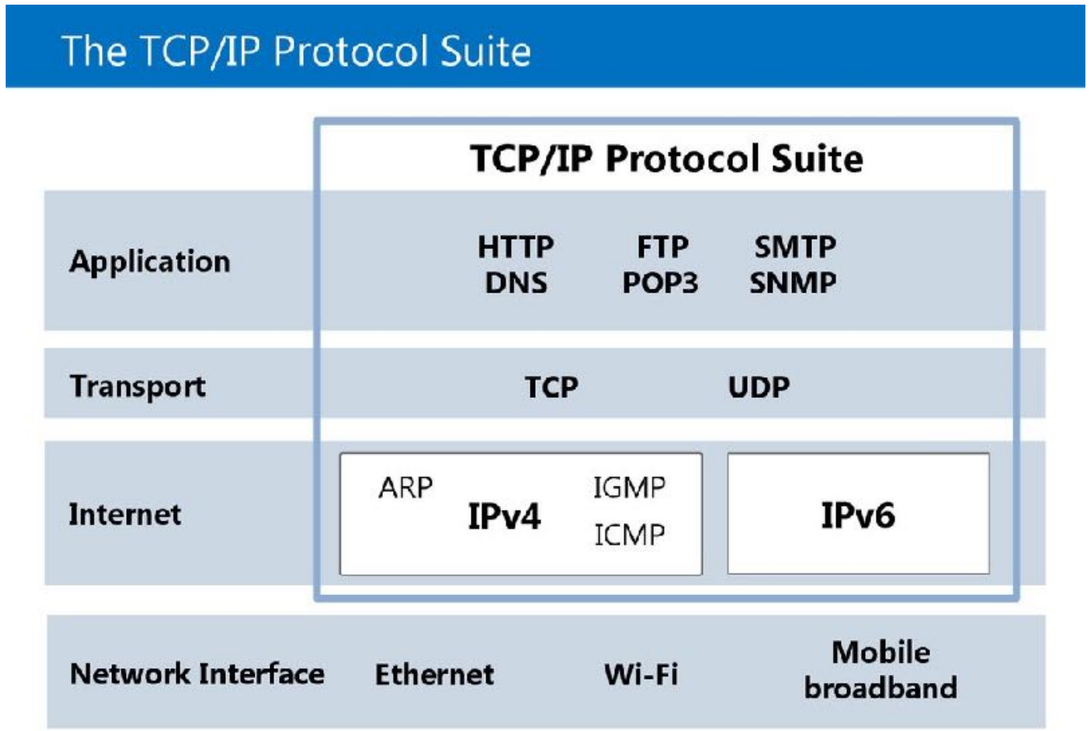
→ HTTP 1.1

→ HTTP 2.0 / SPDY

→ HTTP 3.0 / QUICK

Download slides: lvl.li/htmlheaders

HTTP within TCP/IP



Download slides: lvl.li/htmlheaders

Why secure it browser side?

- → Session hijacking
- → Cookie theft
- → Account takeover
- → Redirecting traffic
- → Stealing account credentials
- → Displaying unwanted ads
- → Virus/malware infections
- → Keylogging

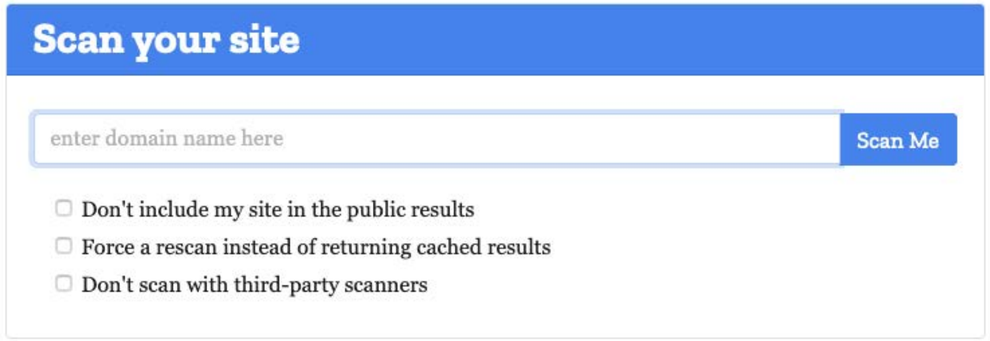
Download slides: lvl.li/htmlheaders

Source: Miriam Schwab at WordCamp US 2018

How to evaluate current headers?

→ https://securityheaders.com

→ https://observatory.mozilla.org



Download slides: lvl.li/htmlheaders

Testing with local tools

→ I love **curl**: $ curl -I https://www.example.com/

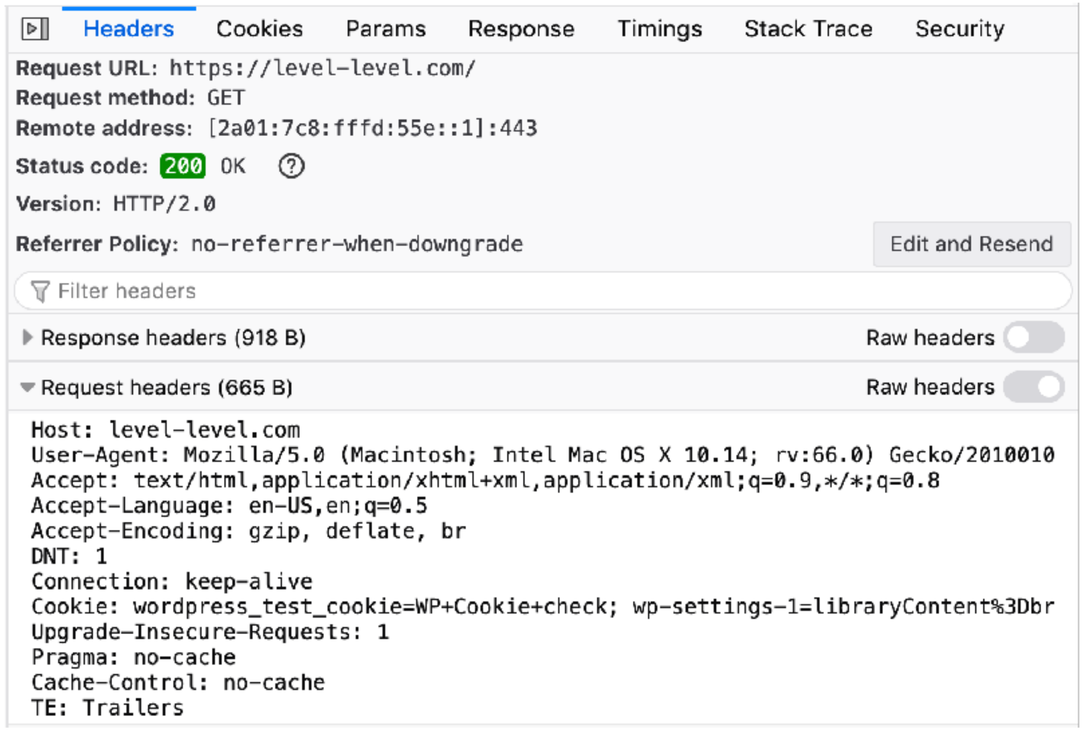
→ $ apt install curl || brew install curl

→ Others:

- Wget
- Network Tab from Developer Tools

Download slides: lvl.li/htmlheaders

A HTTP Request



Download slides: lvl.li/htmlheaders

A HTTP Reply Message

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 05 Apr 2019 15:32:14 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Strict-Transport-Security: max-age=63072000
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-Xss-Protection: 1; mode=block
Content-Security-Policy: default-src https: data: 'unsafe-inline' 'unsafe-eval'
X-Jobs: Looking for a Job? Get in touch: mail (at) level-level (dot) com
Access-Control-Allow-Origin: https://level-level.com
Access-Control-Allow-Methods: GET, POST, OPTIONS

<!doctype html>
<html class="no-js" lang="en-US" prefix="og: http://ogp.me/ns#">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Level Level - Full Service WordPress Agency - Rotterdam</title>

<!-- This site is optimized with the Yoast SEO Premium plugin  - https://yoast.com/wordpress/plugins/seo/ -->
<meta name="description" content="Level Level is known for bespoke digital experiences that make your brand stand out, are fully
accessible, easy to use and run smoothly even under pressure."/>
<link rel="canonical" href="https://level-level.com/" />
```

Bla bla bla... I deleted all caching and non relevant headers. Sorry for that.

Download slides: lvl.li/htmlheaders

Bad?! HTTP headers

```
$ curl -I https://www.example.com/

HTTP/1.1 200 OK
Server: Apache/2.4.25 (Unix) OpenSSL/1.0.1e-fips
X-Powered-By: PHP/7.1.15
Date: Fri, 05 Apr 2019 15:00:28 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Vary: Accept-Encoding
Link: <https://example.com/wp-json/>; rel="https://api.w.org/"
X-Pingback: https://example.com/xmlrpc.php
```

Download slides: lvl.li/htmlheaders

HTTP & TLS

→ TLS version 1.2

→ Ciphers, use AES 128 and 256 bit

→ Key exchange, use Elliptic-Curve Diffie-Helman (ECDHE)

→ HPKP (HTTP Public Key Pinning) just forget it!

→ Don’t buy EV certificates

→ Let’s Encrypt rocks but can be tricky

Download slides: lvl.li/htmlheaders

Get TLS right.

Aim for an A(+) score on SSLLabs

https://www.ssllabs.com/ssltest/

Download slides: lvl.li/htmlheaders

```
TLS_ECDHE_RSA_AES128_GCM_SHA256
TLS_ECDHE_RSA_AES256_GCM_SHA384
TLS_ECDHE_RSA_AES128_CBC_SHA256
TLS_ECDHE_RSA_AES256_CBC_SHA384
```

SSLLabs reports are a bit cryptic.

That's the way cryptologists like it.

Download slides: lvl.li/htmlheaders

HSTS

→ HSTS or HTTP Strict-Transport-Security

→ Age must be 60 days or more

→ IncludeSubDomains

→ Preload

Strict-Transport-Security: max-age=2592000

Download slides: lvl.li/htmlheaders

Expect-CT header

```
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
```

→ Chrome's Certificate Transparency policy

→ Open framework for monitoring and auditing SLL certificates

→ Restore trust in CA's (DigiNotar)

→ IETF draft

→ Let's Encrypt supports it

→ https://scotthelme.co.uk/a-new-security-header-expect-ct/

Download slides: lvl.li/htmlheaders

Minimal Viable Headers

X-Frame-Options: DENY

→ Deny others from framing your content

→ Start with DENY

→ SAMEORIGIN, ALLOW-FROM https://example.com

Download slides: lvl.li/htmlheaders

Minimal Viable Headers

X-Xss-Protection: 1; mode = block

→ Enable browsers built in XSS protection

→ Firefox does not support it

→ Content-Security-Policy “frame-src” will replace it

Download slides: lvl.li/htmlheaders

CORS Header

Cross-Origin Resource Sharing

→ Access-Control-Allow-Origin: https://level-level.com

→ Access-Control-Allow-Methods: GET, POST, OPTIONS

Download slides: lvl.li/htmlheaders

CSP Header

```
Content-Security-Policy: default-src 'self';
font-src 'self' 'unsafe-inline' data: https://
fonts.gstatic.com; img-src 'self' data: https://
secure.gravatar.com; script-src 'self' 'unsafe-
inline' data: https://www.googletagmanager.com
https://consent.cookiebot.com; style-src 'self'
'unsafe-inline' data: https://
fonts.googleapis.com; block-all-mixed-content;
upgrade-insecure-requests;
```

Download slides: lvl.li/htmlheaders

More on CSP

→ Test and test it again. Use report-only mode

→ Script-src ‘unsafe-inline’, this is considered a unsafe practice

→ Miriam Schwab’s talk on CSP (WC EU or US 2018)

→ Matt Brunt: CSP: Let’s Break Stuff (WC LDN 2018)

→ Use a tool like https://report-uri.com/home/tools

Download slides: lvl.li/htmlheaders

Feature-Policy header

```
Feature-Policy: accelerometer 'none';
ambient-light-sensor 'none'; autoplay
'none'; camera 'none'; fullscreen
'none'; geolocation 'self'; gyroscope
'none'; magnetometer 'none'; microphone
'none'; midi 'none'; speaker 'none';
sync-xhr 'self'; usb 'none'; vr 'self'
```

Source: https://scotthelme.co.uk/a-new-security-header-feature-policy/

More on Feature-Policy

→ Not all proposed directives are implemented, like gyroscope, vibrate, notifications, speaker

→ Chrome / Opera dev console shows errors on the “payment” directive

→ securityheaders.com shows an error “invalid cookie directive”

→ Firefox and other no-chrome browsers do not support Feature-Policy, yet.

→ Protocol feels like a Beta at best

Download slides: lvl.li/htmlheaders

Go home and play! 

For testing all the different HTTP headers from within WordPress I found these plugins:

https://wordpress.org/plugins/http-security/

https://wordpress.org/plugins/http-headers/

Download slides: lvl.li/htmlheaders

Good to know when testing

→ Things will and should break

→ Deactivate privacy & ad-blockers browser plugins

→ Deactivate security and caching plugins

→ Nginx or Apache should not add any HTTP headers

→ Nginx with fastcgi, don’t hide and ignore headers

→ If headers don’t show, you probably miss some quoting of values

→ WordPress default sends some headers

Download slides: lvl.li/htmlheaders

Production

→ Implement all or most HTTP headers on the Nginx or Apache HTTP server

→ When a website is behind a proxy/loadbalancer:

- Be careful not to override or double headers
- Define headers on the proxy as much as possible
- Caching headers should be defined on the origin

Download slides: lvl.li/htmlheaders

Thank you!

Slides & links on: lvl.li/htmlheaders

Find us on:

@levellevel

Links

- → https://developer.mozilla.org/en-US/docs/Web/HTTP
- → https://scotthelme.co.uk
- → https://certbot.eff.org/
- → https://www.owasp.org/
- → https://www.owasp.org/index.php/OWASP_Secure_Headers_Project

Download slides: lvl.li/htmlheaders