Security and Insecurity of HTTP Headers

Dirk Wetter

@drwetter

myself

- Independent Security Consultant
  - Offense / Defense / Security Project Management
  - Historical strong Unix/networking background
- OWASP Involvement
  - OWASP AppSec Research 2013
  - German OWASP Day 2012, 2014

  } chair
  - German Chapter Lead
  - Helping hand here/there
- Other
  - My TLS hobby: testssl.sh

Security and Insecurity of HTTP Headers

Disclaimer

- ~~Completeness~~
- Perspective

1. What, (in)security?

- Instruction: Browser
  - Do this
    - 30x, 401
    - Cache
    - Expire
    - Do not frame this, set Cookie, keep-alive, etc
  - Displaying: specify Content-Type/Encoding, Compression
- More properties to identify session / keep backend server

1. What, (in)security?

- Redundant information
  - Type of server, application
  - Version numbers, more or less
  - Framework
  - Architecture hints
    - Via <proxy>
    - IP of load balanced server

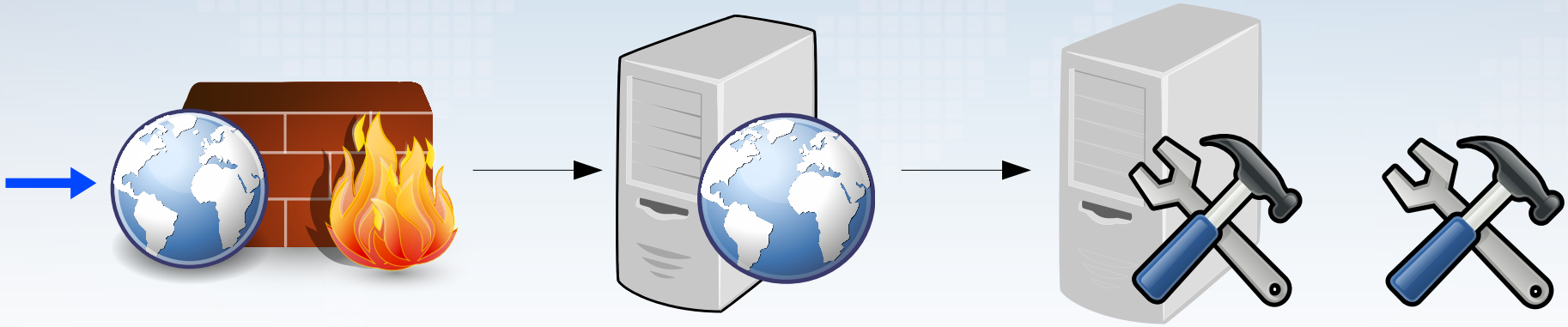
2. Where does it come from?

- Where do header lines originate from?
  - Simple setup

(Web+Appl server} + Application

2. Where does it come from?

- Where do header lines originate from?
  - Not so simple setup



RP or WAF Entry web server App server + Application

2. Where does it come from?

<OT> Often forgotten

- Server time!
  - HTTP header: Date
  - TLS timestamp
    - SChannel: all
    - OpenSSL < 1.0.1f
  - 1-4 conclusions from date timestamps

</OT>

```
Testing now (2015-05-16 22:44) --->[redacted]:443 ([redacted]) <---

 rDNS ([redacted]):  --
 Service detected:       HTTP


--> Testing server defaults (Server Hello)

 TLS clock skew:              +159 sec from localtime
 HTTP clock skew:             +20 sec from localtime
 TLS server extensions        server name, renegotiation info, session ticket
 Session Tickets RFC 5077     (none)
 Server key size              2048 bit
 Signature Algorithm          SHA256withRSA
 Fingerprint / Serial         SHA1 [redacted]DB1D92056B628EE26345E4AB4[redacted] / [redacted]9988
                              SHA256 [redacted]49CE2D445F9C8C4892D8018B948E0F9F74859F[redacted]
 Common Name (CN)             [redacted] (works w/o SNI)
 subjectAltName (SAN)         [redacted]
 Issuer                       thawte SSL CA - G2 (thawte, Inc. from US)
 Certificate Expiration       >= 60 days  (2015-01-26 01:00 --> 2018-04-27 01:59 +0200)
 # of certificates provided   2
 Certificate Revocation List  http://tj.symcb.com/tj.crl
 OCSP URI                     http://tj.symcd.com
 OCSP stapling                not offered


Done now (2015-05-16 22:44) --->[redacted]:443 ([redacted]) <---
```


2. Where does it come from?

- Further Interesting stuff
  - Big IP F5

    `Set-Cookie: BIGip<str>=1677787402.36895.0000;`
  - Netweaver-Cluster

```
dirks@laptop:~|0% BIG-IP_cookie_decoder.py 1677787402.36895.0000

[*] String to decode: 1677787402.36895.0000

[*] Decoded IP:   10.1.1.100
[*] Decoded port: 8080

dirks@laptop:~|0%
```


2. Where does it come from?

- Conclusion #I
  - Fingerprinting of
    - Architecture / Infrastructure
    - Patchlevel of Components
    - Maybe more (see l8er)
  - Evil dude: Gimme more!

2. Where does it come from?

- Conclusion #II
  - Client side perspective:
    - What's *really* necessary?
    - What else can be done to improve security?

```
dirks@laptop:~|1% wget -qSO /dev/null eiklaut.net
  HTTP/1.1 200 OK
  Date: Thu, 21 May 2015 22:44:42 GMT
  Content-Type: text/html
  Content-Length: 630
  Last-Modified: Fri, 29 Nov 2013 08:39:54 GMT
  Server: Apache 2.2.22 (RHEL), mod_ssl openssl 1.0.3
  Set-Cookie: PHPSESSID=44h5vc482fgiapfmit5t0n9f03; path=/;
  X-Powered-By: PHP/4.4.42
  X-UA-Compatible: IE=EmulateIE7
  Accept-Ranges: bytes
  Connection: keep-alive
dirks@laptop:~|0%
```

2. Where does it come from?

- Conclusion #II :

  - If / where possible:
    - **Proper secure defaults!**
    - Remove chatty lines
    - add security headers

3. Theory

- Cookie attributes / flags
  - `HttpOnly`
    - access of JavaScript methods → cookie
    - classical XSS
      - some browser protection nowadays
  - `Secure`
    - no transport via plaintext HTTP
      - cookie clobbering, mixed content
      - SOP, works?!
        - » Some bypasses

3. Theory

- Transport encryption: HSTS (RFC 6797)
  - HTTP `Strict-Transport-Security`
  - @Browser : Within `max-age` never ever use plain http, maybe `includeSubDomains`
  - `max-age:` [seconds]
    - Recommended > ½ year: 3600 • 24 • 181
  - Browser support ok *)
    - ✔ Chrome >=4
    - ✔ Firefox >= 4
    - ✔ Safari >=7, Opera >=12.1
    - • IE 11 on W10 only!

*) see http://caniuse.com/#feat=stricttransportsecurity

3. Theory

- Transport encryption: HSTS
  - One problem though

TOFU!

  - `preload:`
    http://hstspreload.appspot.com/

3. Theory

- Transport encryption: HSTS
  - Pitfalls
    - Change of mind (http!) within time specified
    - Certificate expired (example Firefox)
    - Careful with `includeSubDomains` for TLD!

This Connection is Untrusted

You have asked Firefox to connect securely to **owasp.de**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

This site uses HTTP Strict Transport Security (HSTS) to specify that Firefox only connect to it securely. As a result, it is not possible to add an exception for this certificate.

Get me out of here!

Technical Details

owasp.de uses an invalid security certificate.

The certificate expired on 04/28/2015 09:58 AM. The current time is 04/29/2015 12:37 AM.

(Error code: sec_error_expired_certificate)

3. Theory

- Transport encryption: HSTS
  - Solution to Pitfalls
    - set `max-age` to 0 (see RFC 6797, 6.1.1)
      - worked w/ FF
      - Chrome?
    - order new certificate ;-)

3. Theory

- IETF ~~Draft~~ RFC 7469

  HTTP `Public-Key-Pins` / `Public-Key-Pins-Report-Only`
  - Certificate pinning a.k.a. HPKP
  - @Browser: Within `max-age` remember keys and do not except anything else
  - `includeSubDomains`
  - `pin-sha256=<base64str>` …
  - `report-uri=<json-POST-URI>` !

3. Theory

- Goodbye (un)lawful / whatsoever interception ;-)
  - RFC: **UAs MUST close the connection to a host upon Pin Failure**

- Ok, some constraints...
  - (quality of encryption)
  - Local CAs
  - TOFU

Picture courtesy of Submessenger @ www.bullshido.net

3. Theory

- HPKP: which keys?
  - Best: pins of >= two keys
    - actual
    - Pin of backup key in pocket / safe
    - Careful with issuer pinning
  - how? script from Hanno Böck
  - Locking out?
    - `max-age=0` ???

3. Theory

```
dirks@laptop:~|0% wget -qS -O /dev/null https://testssl.sh
  HTTP/1.1 200 OK
  Date: Sun, 17 May 2015 17:15:27 GMT
  Content-Type: text/html; charset=UTF-8
  Transfer-Encoding: chunked
  Connection: keep-alive
  Server: Never trust a banner
  X-Powered-By: A portion of humor
  Public-Key-Pins: pin-sha256=0SAMpcsNkPtjORdHdRDxho0NjSUJBgJGVPIFfieSEeA=;
pin-sha256=WvwV39oyQzKmrc1cC5CU7NImVeJlbGZ/mwAnfwMpNOw=; max-age=2592000
  Strict-Transport-Security: max-age=31337000
  X-FRAME-OPTIONS: DENY
  X-Content-Type-Options: nosniff
  X-XSS-Protection: 1; mode=block
dirks@laptop:~|0% openssl s_client -connect testssl.sh:443 -servername testssl.
sh 2>/dev/null </dev/null | \
awk '/-----BEGIN/,/-----END/ { print $0 }' | \
openssl x509 -pubkey -noout | \
grep -v PUBLIC | \
openssl base64 -d | openssl dgst -sha256 -binary | openssl base64
WvwV39oyQzKmrc1cC5CU7NImVeJlbGZ/mwAnfwMpNOw=
dirks@laptop:~|0%
```

3. Theory

- HPKP, browser support:
  - ✔ Chrome >=35
  - ✔ Firefox >= 35
  - Safari, Opera?
  - IE 1x probably not at all (yet)

https://projects.dm.id.lv/Public-Key-Pins_test

3. Theory

Was: per site

Now per page

3. Theory

- `X-Content-Type-Options: nosniff`
  - Originally designed for IE < 8
    - MIME sniffing! Ignored Content-Type completely
      - Similar unix `file` (or MIME libs → file uploads)
    - User controlled content
      - chameleon files, picture XSS
      - HTML/JS in PS, PNG etc.
    - Attention:
      - Compatibility view of IE

3. Theory

- Safer dl'ing:
  - `Content-Disposition: attachment`
  - `X-Download-Options : noopen`

3. Theory

- `X-Content-Type-Options: nosniff`
  - Border cases for Firefox
    - General: trust MIME type
    - What if empty? → PDF/PS, XML, HTML
    - Determine image type

OWASP AppSecEU 15
Amsterdam, The Netherlands

3. Theory

- `X-FRAME-OPTIONS`
  - `DENY | SAMEORIGIN | ALLOW-FROM` *`uri`*
  - Clickjacking 'n friends
  - Against framing **your** page

  - Attention:
    - Business reasons for framing
    - Application (e.g. Typo3 backend)
    - Chrome/Webkit? don't give a damn: `ALLOW-FROM`

3. Theory

- XSS
  - `X-XSS-Protection`
    - `0` → off, sense? (https://www.facebook.com/)
    - `1` → on, browser engine can sanitize
    - `1; mode=block` → better: no rendering
    - Good thing (again):
      - `report=<JSON POST URI>`
  - Support:
    - no Firefox
    - ✔ Chrome/Webkit
    - ✔ IE

3. Theory

```
dirks@laptop:~|0% wget -qS -O /dev/null 'https://www.youtube.com/supported_browsers?next_url=%2F'
  HTTP/1.1 200 OK
  Date: Sun, 17 May 2015 12:52:51 GMT
  Server: gwiseguy/2.0
  Content-Type: text/html; charset=utf-8
  X-Content-Type-Options: nosniff
  Expires: Tue, 27 Apr 1971 19:44:06 EST
  Cache-Control: no-cache
  X-Frame-Options: SAMEORIGIN
  X-XSS-Protection: 1; mode=block; report=https://www.google.com/appserve/security-bugs/log/youtube
  P3P: CP="This is not a P3P policy! See http://support.google.com/accounts/answer/151657?hl=de for
  Set-Cookie: VISITOR_INFO1_LIVE=E9cKrYF0ABY; path=/; domain=.youtube.com; expires=Sat, 16-Jan-2016
  Set-Cookie: PREF=f1=50000000; path=/; domain=.youtube.com; expires=Sat, 16-Jan-2016 00:45:51 GMT
  Set-Cookie: YSC=sRu7VBfLE4Q; path=/; domain=.youtube.com; httponly
  Set-Cookie: hideBrowserUpgradeBox=True; path=/; domain=.youtube.com; expires=Sun, 31-May-2015 12:5
  Alternate-Protocol: 443:quic,p=1
  Accept-Ranges: none
  Vary: Accept-Encoding
  Transfer-Encoding: chunked
dirks@laptop:~|0%
```


3. Theory

`X-Content-Security-Policy` (FF < 23)
`X-WebKit-CSP` (Chrome < 25)

- XSS, again
  - `Content-Security-Policy` (+ related)

    (`Content-Security-Policy-Report-Only`)
    - not trivial!
      - Cooperation of development, JS frameworks, templates, trackers, objects embedded, ...
    - ~two parts
      1. **Policy directive** (*-src)
      2. **source value(s)**
    - Concat as much pairs as you want
      - Separation by semicolon

3. Theory

- XSS, again
  - `Content-Security-Policy`
    - Policy directive

      @Browser: which content are you allowed to render?

      - » `script-src`
      - » `img-src`
      - » `style-src`
      - » `frame-src` (father option to X-FRAME-OPTIONS)
      - » `font-src`
      - » `media-src` (video, audio)
      - » `default-src`
      - » ...

3. Theory

- XSS, again
  - `Content-Security-Policy`
    - Source value: @Browser: this are you allowed to do with directive
      - `'none'`
      - `'self'`
      - `'unsafe-inline'` → Attention: standard defense XSS
      - `'unsafe-eval'` → bad
      - *Uri*
        - „trick“ only `https:`
        - `*` → Attention!
  - Violations (Json POST URI):
    - `Content-Security-Policy <key value> report-uri <URI>`
  - Support
    - Chrome/ Firefox: good!
    - IE 10/11 fragmentary
  - Shameless plug: OWASP TT 2013

See https://www.owasp.org/index.php/Content_Security_Policy_Cheat_Sheet

3. Theory

```
dirks@laptop:~|0% wget -qS -O /dev/null https://github.com
  HTTP/1.1 200 OK
  Server: GitHub.com
  Date: Sun, 17 May 2015 13:22:05 GMT
  Content-Type: text/html; charset=utf-8
  Transfer-Encoding: chunked
  Status: 200 OK
  Content-Security-Policy: default-src *; script-src assets-cdn.github.com collector-cdn.
com; object-src assets-cdn.github.com; style-src 'self' 'unsafe-inline' 'unsafe-eval' ass
.github.com; img-src 'self' data: assets-cdn.github.com identicons.github.com www.google-
cs.com collector.githubapp.com *.githubusercontent.com *.gravatar.com *.wp.com; media-src
; frame-src 'self' render.githubusercontent.com gist.github.com www.youtube.com player.vi
 checkout.paypal.com; font-src assets-cdn.github.com; connect-src 'self' live.github.com
ive.github.com uploads.github.com status.github.com api.github.com www.google-analytics.c
ub-cloud.s3.amazonaws.com
  Cache-Control: no-cache
```


3. Theory

- CSP v2 (still W3C draft)
  1. Whitelist scripts: Hash support
     - Inline Code:
```
<script>
    notevilstuff
</script>
```
     - Header: `Content-Security-Policy: script-src 'sha256-<base64 encoded hash(“notevilstuff“))>';`

  2. Whitelist script: Nonce
     - Inline Code:
```
<script nonce='n0n53'>
    notevilstuff
</script>
```
     - Header: `Content-Security-Policy: script-src 'nonce-n0n53';`

3. Theory

- CSP v2 (still W3C draft)
  3./4. Same whitelists with `style-src`
  5. `frame-ancestors`

4. Where to set / unset?

- Depends...
  - Your access
  - Your hat on your head
- In principle 4 possibilities
  - application
  - application server configuration

    `{server,web}.xml`, `web.config`, `php.ini` etc.
  - web server (configuration)
  - reverse proxy / load balancer / WAF

4. Where to set / unset?

- What's the best place?
  - KISS
    - What ever is the easiest for you
    - As long as the application still works
  - Best
    - At the root of the cause

4. Where to set / unset?

- Disclaimer: Obscurity & Security
  - Do PROPER defense!
  - Maybe protection against shodan search
  - Otherwise: criminals throw everything @ target

4. Where to set / unset?

- Pitfalls
  - Application needs still to work!
    - Cookie: Secure, HTTPOnly, restricting domain scope
    - Caching

5. Practice

- Two examples

  Looking @ Web servers (reverse proxy, resp.)

  – Apache + nginx
    - no IIS today

  – Tasks
    1. Minimize default verbosity
    2. Provide / remove additional headers

5. Practice

- Apache
  - Default configuration
    - `Server: Apache/2.2.3 (Linux/SUSE)`
    - `Server: Apache/2.4.8 (Win32) OpenSSL/1.0.1b PHP/5.5.3`
    - `Server: Apache/2.2.17 (Ubuntu)PHP/5.3.5-1ubuntu7.5 with Suhosin-Patch mod_python/3.3.1 Python/2.7.2`

I. Secure defaults

- `ServerTokens Prod` (Header) while you're at it: `ServerSignature Off` (Error)
- Results in `Server: Apache`
- not enough?

```
a2enmod security2; echo \
  "ServerTokens Full\nSecServerSignature Microsoft-IIS/8.0" \
>>  /etc/<somepath>/mod*security*.conf
```

- omit “Server:” not possible (recompile)

5. Practice

- Apache
  - II. Remove /add header lines
    - a2enmod headers
      - `Header always set X-FRAME-OPTIONS sameorigin`
      - `Header always set X-Content-Type-Options nosniff`
      - `Header always set X-XSS-Protection "1; mode=block"`
      - `Header always set Strict-Transport-Security "max-age=16070400" env=HTTPS`
      - `Header edit Set-Cookie "^(.*)" $1;Secure env=HTTPS`
      - `Header edit Set-Cookie "^(.*)" $1; HttpOnly`
      - `Header unset "X-Pingback"`
      - `Header always unset "X-Powered-By"`
      - `Header unset sap-cache-control`

5. Practice

- nginx
  - Default configuration
    - `Server: nginx/1.X.Y`

  I. Secure defaults
    - remove banner (both error and header):
      - `server_tokens off;`
    - Enable module: `ngx_headers_more`
      - `more_set_headers  "Server: OWASP ru135";`
      - `more_clear_headers "Server: *";`

  II. Remove/add header lines
    - `more_set_headers`
    - `more_clear_headers`

5. Practice

- Remarks
  1. As usual for nginx/Apache:
     - set/clear headers: per location possible!

     → small “fixes” possible w/o access to app

```
dirks@laptop:~|0% curl -si  -o - php.net | grep X-Powered-By
X-Powered-By: PHP/5.6.7-1
dirks@laptop:~|0%
```

  2. PHP
     - Better @ r00t of all evil
       - e.g. `php.ini: expose_php=off`
     - Same: real app server hardening

5. Practice

- Test it!
  - Tool from the outside
    - curl / wget / devel tools browser
  - Thorough functional tests!
    - **BROWSERS!!**

```
dirks@laptop:~|0% curl -si appsec.eu
HTTP/1.1 302 Found
Date: Thu, 21 May 2015 10:12:53 GMT
Server: Cool OWASP Conference 2015
X-FRAME-OPTIONS: DENY
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Pect: 2016 AppSec EU conference
Location: http://2015.appsec.eu/
Vary: Accept-Encoding
Content-Length: 206
Content-Type: text/html; charset=iso-8859-1
```