Dirk Wetter

Kleine, feine Helferlein: HTTP Header und Security

OWASP

The Open Web Application Security Project

About Me

- Einzelunternehmer
  - Beratung Web-/Sicherheit (http://drwetter.eu/neu/)
    - Überprüfen / Absichern / Beraten+Schulen
- Ehrenamt, zuviel ;-)
  - OWASP AppSec Research 2013
  - German OWASP Day 2012, 2014
  - OWASP-u.a. Programmkomitees
  - GUUG (ggw. Interimsvorstandsvorsitzender)

Agenda

- Was ist das?
- Sicherheitsrelevanz?
- Welche
  - Theorie
  - Praxis

Was ist das?

- Anweisung: Browser
  - Direkt was zu tun (30x, Keep-Alive)
  - Indirekt was zu tun oder zu lassen
    - Seite framen/nicht, Cookie, Caching etc
  - Darstellungsebene
    - Content-Type/Encoding, Compression, ...
    - Skripte
- Redundante Informationen
  - Server, weitere Komponenten
  - Applikation / Framework
- Nicht redundante Informationen
  - Serverzeit

Was ist das? / Beispiele

myprompt |0% curl -Lis php.net | head -20

HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Mon, 27 Oct 2014 11:08:44 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.5.11-2
Last-Modified: Mon, 27 Oct 2014 18:01:40 GMT
Content-language: en
X-Frame-Options: SAMEORIGIN
Set-Cookie: COUNTRY=NA%2C2001%7A6f8%3A1d37%3A10%3A%312; expires=Mon, 03-Nov-2014 11:08:44 GMT; Max-Age=604800; path=/; domain=.php.net
Set-Cookie: LAST_NEWS=1414408124; expires=Tue, 27-Oct-2015 11:08:44 GMT; Max-Age=31536000; path=/; domain=.php.net
Link: <http://php.net/index>; rel=shorturl
Vary: Accept-Encoding

Sicherheitsrelevanz

- Banner
  - Server, Applikation, Framework
    - Software selber und Versionsnummer
  - Manchmal interne Infos (z.B. IP-Adressen)
    - F5 BigIP-Cookie, Netweaver-Cluster, andere Cluster
    - Vorsicht: Notwendigkeit!
  - Disclaimer: Obscurity & Security
    - Anekdote
- Cookie
  - Pfad, Domain, Gültigkeit, Secure, HTTPOnly
- Caching

Wo setzen/entfernen?

- Prinzipiell 4 Stellen:
  - Applikation →
  - Application Server
    `{server,web}.xml`, `web.config`, `php.ini` etc.
  - Webserver → Praxisteil
  - Reverse Proxy, WAF

```
Java
    • Cookie.setHttpOnly();
    • Cookie.setSecure(boolean);
    • HttpServletResponse.setHeader
    • ..
PHP
    • Header();
    • HttpResponse::setHeader()
    • ...
...
```

Wo setzen/entfernen?

- Wo *soll* man setzen?
  - Je nach Kontext
  - Applikationsspezifische Einstellungen
    → möglichst nahe Applikation (Cookie+Caching)
  - Je weiter Richtung Webserver / Reverse-Proxy:
    → Härtungen

- Cookie-Attribute/Flags
  - `HttpOnly`
    - Zugriff(JavaScript-Methoden) → Keks
  - `Secure`
    - Keine Übertragung via HTTP
    - SOP?!
    - cookie clobbering, mixed content

- Transportverschlüsselung: HSTS
  - Gegen SSL stripping
  - `Strict-Transport-Security`

    Weist den Browser an, im Zeitraum `max-age` niemals diese Seite (ggf. inkl. Subdomains: `includeSubDomains`) unverschlüsselt zu besuchen
  - Zeit in Sekunden
    - Mindestens halbes Jahr: 3600 • 24 •183
  - TOFU / `preload:` http://hstspreload.appspot.com/

→ Demo: https://testssl.sh

Theorie

- Neuer Draft: `Public-Key-Pins(-Report-Only)`
  - Zertifikat-Pinning, HPKP
  - Weist den Browser an, im Zeitraum `max-age` (Sekunden) sich das/die Zertifikate zu merken und derweil kein anderes zu akzeptieren
  - `IncludeSubDomains`
  - `Report-uri`
  - ~~`pin-sha1=`~~, `pin-sha256= ;` (mehrere)

  - *Goodbye (un)lawful interception* ;-)
    - Ok, mit Einschränkungen....
    - TOFU
  - Verfügbarkeitsproblem?
    - Remember: Heartbleed?

  → https://slo-tech.com/

Alles, was nun kommt: per „Seite“

Theorie

- `X-Content-Type-Options`
  - Ursprünglich zur Absicherung alter IE (<8)
    - Misstrauten Content-Type Header komplett
    - Alle anderen Browser sind strikter
  - Grenzfälle, wo andere Browser trotzdem klüger sein wollen
    - Unterschieben von Script-Code
      - Bild-XSS etc
      - Zuvor hochgeladene Dateien (HTML)
    - Mime-Sniffing: Ähnlich Unix Command: `file` (oder diverse MIME-Bibliotheken mit Magic)
    - Option: `nosniff`
    - Nur Chrome/Webkit und IE >=9
  - BTW falls runterladen besser (@Developer_:
    - `Content-Disposition: attachment`

- Clickjacking und Freunde
  - Schutz gegen Seite framen
    - kein Schutz allerdings gegen div-overlays aka stallowned attacks (Rsnake)
  - `X-FRAME-OPTIONS`
    - `DENY | SAMEORIGIN | ALLOW-FROM` *`uri`*
  - Vorsicht:
    - Businessgründe fürs Framen
    - Applikationen (z.B. Typo3-Backend)
    - Chrome/(Webkit?) pfeift auf ALLOW-FROM

→ appsec.eu

Theorie

- XSS, reloaded
  - X-XSS-Protection
    - `0` → Aus, i.d.R. Sinnlos
      https://www.facebook.com/
    - `1` → An, Browser-Engine reinigt ggf.
      - `mode=block`
        → Besser: Browser-Engine rendert Seite ggf. nicht
      - `report=http://site.com/report`
        → schön zu wissen: https://www.xing.com/
  - Haken: Chrome/Webkit, IE, kein Firefox

Theorie

OWASP
The Open Web Application Security Project

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

- XSS, reloaded^2
  - `Content-Security-Policy` (+ Artverwandte)

    (`Content-Security-Policy-Report-Only`)
    - Nicht trivial
      - Zusammenspiel Entwicklung, JS-Frameworks, Tracker, Templates, ...
    - ~Zwei Teile
      1. **Policy-Direktive** (*-src)
      2. **Source List(en)**
         - » Beliebig hintereinander hängen
      - Direktive+Source Liste:
        - » trennen durch Semikolon, falls mehrere

Theorie

- XSS, reloaded^2
  - `Content-Security-Policy`
    - Policy-Direktive

      = Browser: Welchen Content darfst du beziehen?

      » `script-src`

      » `img-src`

      » `style-src` (CSS)

      » `frame-src` (umgekehrt zu X-FRAME-OPTIONS)

      » `font-src`

      » `media-src` (video, audio)

      » `default-src`

      » ...

Theorie

- XSS, reloaded^2
  - `Content-Security-Policy`
    - Teil 2: Source-List

      =Browser: Was darfst du mit dem Content machen
      - `'none'`
      - `'self'`
      - `'unsafe-inline'` → VORSICHT: Standard gegen XSS
      - `'unsafe-eval'` → VORSICHT!
      - *`uri`*
        - » „Trick“ nur `https:`
        - » `*` → VORSICHT
    - Teil 3: *`report-uri`*

    → https://www.github.com

    - Support: Gut, IE 10 und 11 allenfalls lückenhaft
    - Shameless plug: OWASP TT 2013

Theorie

- CSP v2
  - Whitelist scripts (Hash support)
    - Inline Code:
      ```
      <script>
          notevilstuff
      </script>
      ```
    - `Content-Security-Policy: script-src 'sha256-<base64 encoded hash(“notevilstuff“))>';`
  - Nonce
    - Inline Code:
      ```
      <script nonce='n0n53'>
          notevilstuff
      </script>
      ```
    - `Content-Security-Policy: script-src 'nonce-n0n53';`

Praxis

- Nur Webserver / Reverse proxy
  - Apache + nginx
  - ohne mod_security
  - Banner im Header

  - Zwei Aufgaben
    1. Informationsminimierung
       - `X-Powered-By:`
       - `Server: SAP NetWeaver Application Server 7.10 / ICM 7.10`
    2. Zusätzliche Absicherung
       - Siehe Theorie

Praxis

- Apache
  - Basiskonfiguration
    - Banner weg (Fehler und HTTP)
      - `ServerTokens Prod`
      - `ServerSignature Off`
    - `php.ini: expose_php=off`
  - Sonst / zusätzlich
    - **`prompt%`** `a2enmod headers`
      `prompt%`

→ Demo: (Grenzen von mod_headers)

Praxis

- nginx
  - Basiskonfiguration
    - Banner weg (Fehler und HTTP)
      - `server_tokens off`
      - `Php.ini: expose_php=off`
  - Zusätzlich: Module
    - ngx_http_headers_module
      - `add_header`
    - ngx_headers_more
      - `more_set_headers`
      - `more_clear_headers`

  → Demo

OWASP
The Open Web Application Security Project