Reportbasierte CSP-Erzeugung

Einführungsvortrag Projektgruppe

Malte Klaassen

2016-06-16

Inhaltsverzeichnis

Cross Site Scripting

- Angriff auf Nutzer einer Webseite
- Einfügen von Skripts in generierten HTML-Code
  - Reflected XSS
  - Persistent XSS
- Ermöglicht durch unzureichend geprüfte Eingaben

Content Security Policy

- "[...] defines a mechanism by which web developers can control the resources which a particular page can fetch or execute [...]" (CSP Level 3)
- Server sendet Policy in HTTP-Header Content-Security-Policy
- Report/Fetch Directives
- Alternativer Header-Name: Content-Security-Policy-Report-Only

Fetch Directives

- Für XSS insb. relevant: script-src
- Directive-Werte:
  - URIs (ohne Dateiname): https://example.com
  - 'none'
  - 'self'
  - 'unsafe-inline'/'unsafe-eval'
  - 'nonce-VALUE' mit VALUE einem base64-String
  - 'HASHALG-VALUE' mit HASHALG einem Hash-Algorithmus, VALUE dem entsprechenden base64-String
- Neu hinzugeladene Scripts müssen i.A. die Policy ebenfalls erfüllen

Report Directives

- POST von JSON an angegebene URL

```
{"csp-report":
        {"document-uri":
                "http://example.com/index.html"
        ,"referrer":""
        ,"violated-directive":"script-src 'none'"
        ,"effective-directive":"script-src"
        ,"original-policy":
                "script-src 'none'
                ;report-uri http://example.com/
                    collector.php"
        ,"blocked-uri":
                "http://evilsite.com/badscript.js"
        ,"status-code":200}}
```

Policy Beispiel

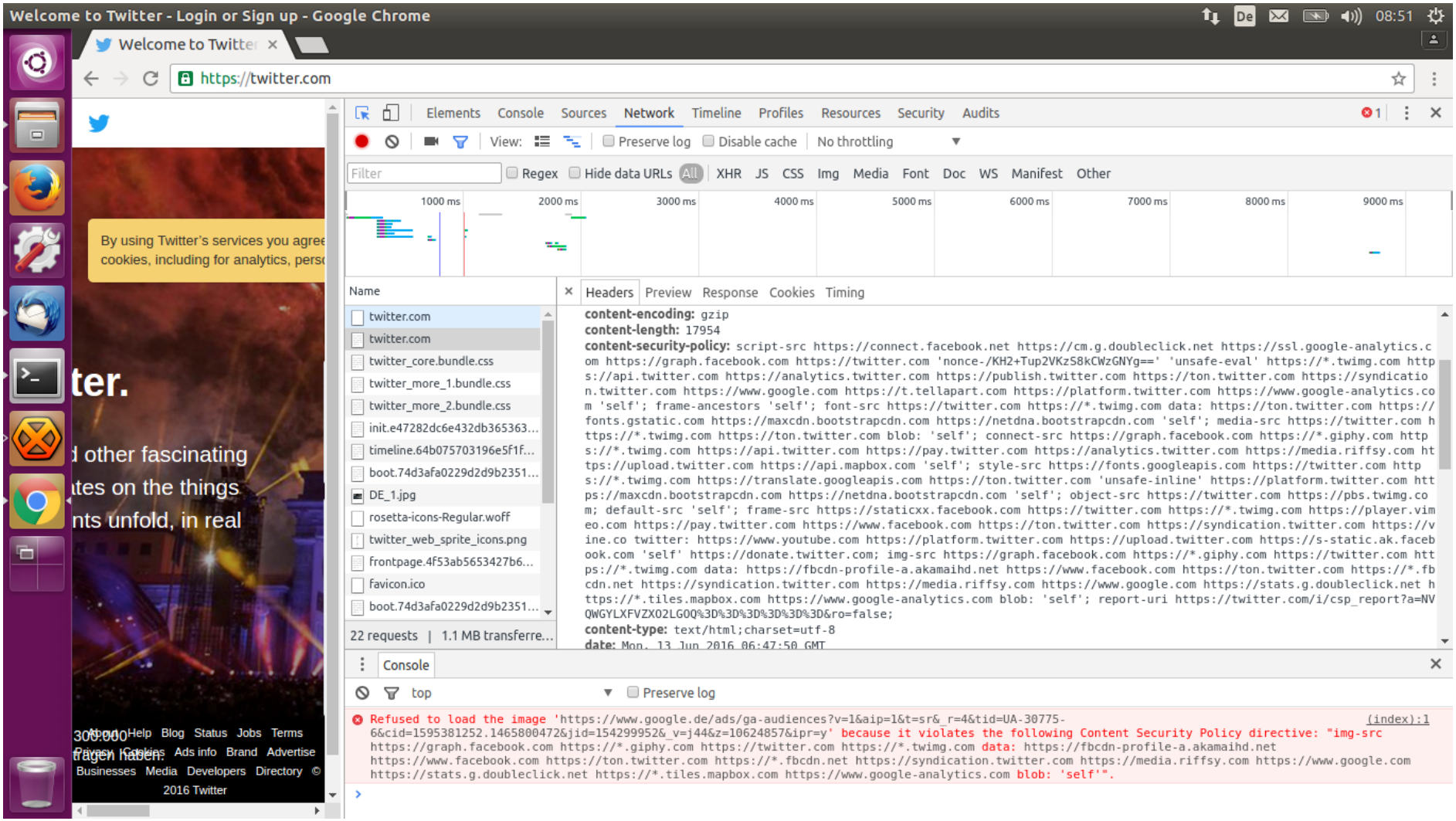


Figure: CSP-Header twitter.com

Probleme der Policy Erzeugung

- Viele verschiedene Ressourcen
- Betreiber kennt nicht alle Ressourcen
  - Erstellung durch WCMS
  - Einbinden durch Fremdskripte
- Ressourcen ändern sich
  - Inhalte hinzugefügt/entfernt
  - Fremdskripte ändern sich

Reportbasierte Policy Erzeugung

- Ziel: Erstellung von Policies aus Reports, unabhängig vom verwendeten Webserver

Policy Erzeugung mit Rückfrage

- Sammlung von Reports, Hinzufügen zur Policy nur nach Rückfrage beim Seitenbetreiber
  - Neue Ressourcen müssen erst freigeschaltet werden, sind zuerst nicht verfügbar
  - Bei Einbindung durch fremde Skripte: Betreiber kennt nicht alle validen Ressourcen
    Kann durch Zurückführung auf ursprüngliche Skripte vereinfacht werden
  - Auwändig/nicht automatisiert

Policy Erzeugung - Beliebige Quellen

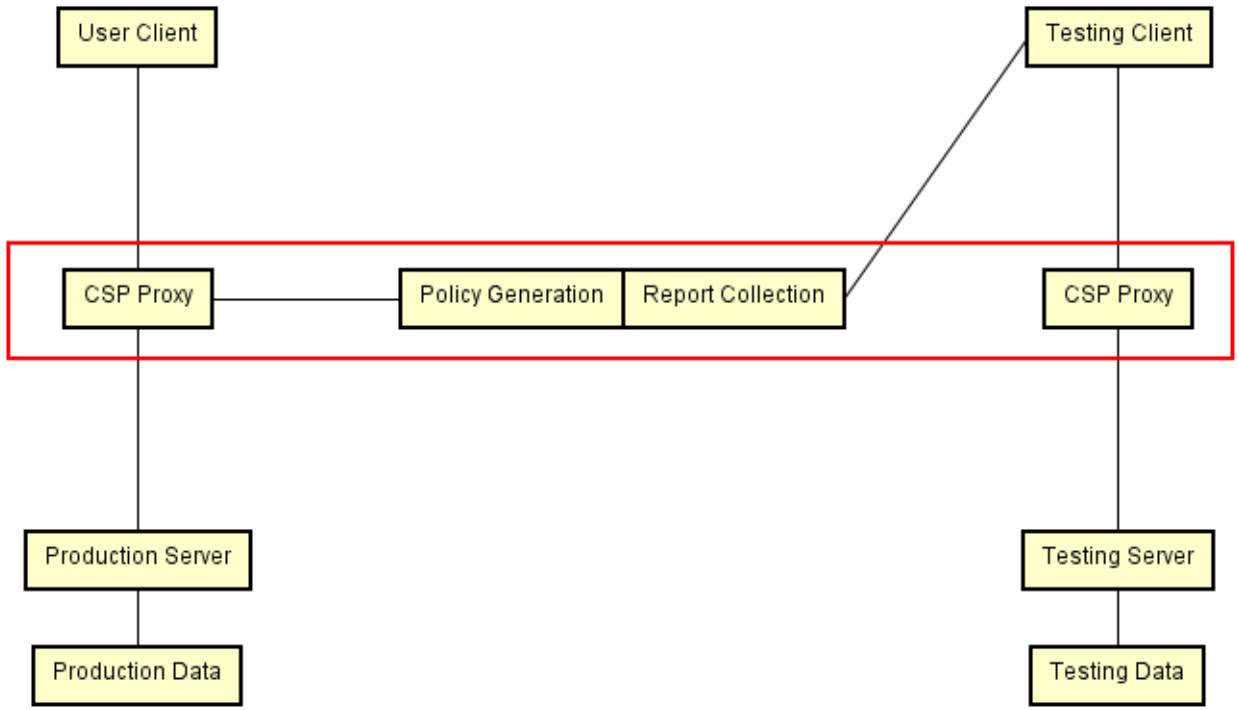
- Vollautomatische Generierung der Policy zur Laufzeit aus Violation Reports von beliebigen Quellen
  - Neue Ressourcen sind zuerst nicht verfügbar
  - Bewertung einer Violation/Ressource nichttrivial nur anhand der Häufigkeit möglich
    $\Longrightarrow$ manipulierbar durch Angreifer

Policy Erzeugung - Sichere Quellen, Produktionsdaten

- Vollautomatische Generierung der Policy aus Violation Reports aus sicheren Quellen auf Produktionsdatensätzen
  - Sichere Quelle : Modul des Proxies, bestehende Tests, ...
  - Quelle muss Kenntnisse über die geproxyte Seite haben
  - Schützt nicht gegen Persistent XSS

Policy Erzeugung - Sichere Quellen, Testdaten

- Vollautomatische Generierung der Policy aus Violation Reports aus sicheren Quellen auf sicheren Testdatensätzen



Policy Erzeugung - Sichere Quellen, Testdaten

- Vollautomatische Generierung der Policy aus Violation Reports aus sicheren Quellen auf sicheren Testdatensätzen
  - Kann bestehendes Testsystem nutzen
  - Schützt auch vor Persistent XSS
  - Policy muss bei jeder Änderung der Ressourcen erstellt werden, auch bei Änderung von Fremdressourcen

Policy Erzeugung - Sichere Quellen

- Proxy
  - Reverse Proxy, fügt Header hinzu
  - Testing-Header: Erlaube nichts, berichte an Collector
  - Production-Header: Von Generator erzeugte Policy
- Collector
  - Nehme Reports entgegen, gebe diese an Generator weiter
- Generator
  - Nehme Reports vom Collector entgegen, erzeuge Policy

TODO