# New Standards and upcoming Technologies in Browser Security

Tobias Gondrom
*Board member of OWASP London*
*Chair of IETF Web Security WG*

tobias.gondrom@gondrom.org

## Tobias Gondrom

- 12 years information security experience
- 10 years application development experience
- Information Security & Risk Management, Research and Advisory, Director
- Author of Standards on Digital Signatures and Secure Archiving
- Chair of IETF Web Security Working Group
  http://datatracker.ietf.org/wg/websec/charter/
  Member of the IETF Security Directorate
- London OWASP chapter board member
  (former OWASP Germany chapter lead)
  www.owasp.org

# Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

## Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

## History

- Internet/Arpanet Protocols were designed for robustness and exchanging information and cross reference  of content…

  …. but not with security and active content in mind

- We try to fix Application Security on the Application end ever since

## Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

# What’s the problem - OWASP Top 10

- A1: Injection
- A2: Cross-Site Scripting (XSS)
- A3: Broken Authentication and Session Management
- A4: Insecure Direct Object References
- A5: Cross Site Request Forgery (CSRF)
- A6: Security Misconfiguration
- A7: Failure to Restrict URL Access
- A8: Insecure Cryptographic Storage
- A9: Insufficient Transport Layer Protection
- A10: Unvalidated Redirects and Forwards

## What’s the problem - OWASP Top 10

*A1: Injection*

*A2: Cross-Site Scripting (XSS)*

A3: Broken Authentication and Session Management

A4: Insecure Direct Object References

*A5: Cross Site Request Forgery (CSRF)*

*A6: Security Misconfiguration*

A7: Failure to Restrict URL Access

A8: Insecure Cryptographic Storage

*A9: Insufficient Transport Layer Protection*

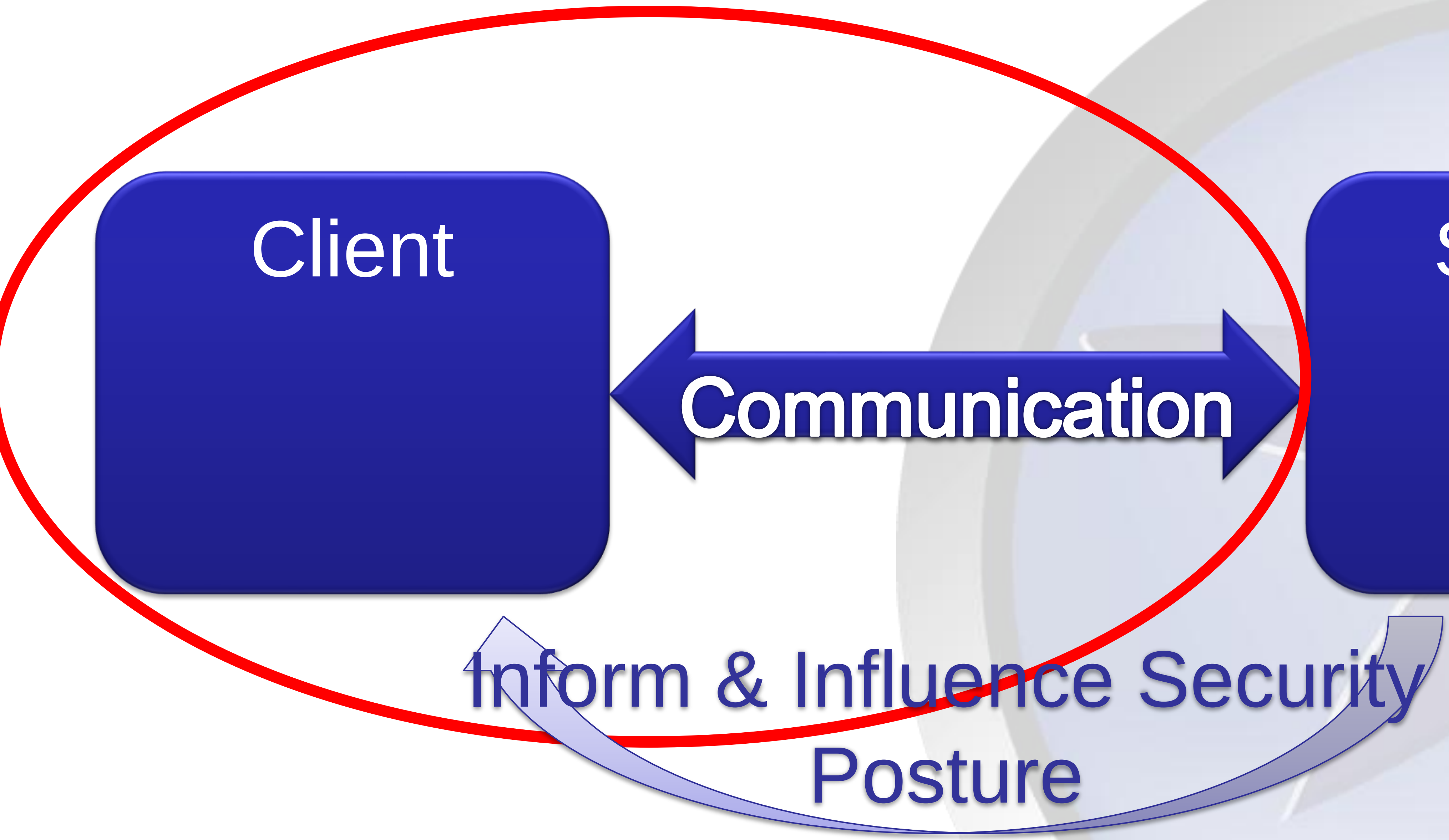
A10: Unvalidated Redirects and Forwards

# What's the problem

- No Clear separation between content and executed code
- Relies on trust relationships (trust on first use / trusted source)
- Weak channel protection
- Authentication & leakage of credentials

=> Today, Web Applications try to fix this on the Application level with little support of the underlying infrastructure

## What’s the problem



## Think Big

- What if we can….

…. improve the underlying infrastructure and protocols?

# Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

# Who – Introducing the Players

- OWASP
  - Top Ten
  - Browser Security Day at OWASP Summit
- IETF
  - Web Security WG
- W3C:
  - HTML5
  - Web App Sec WG
- Browser Vendors

## Why

- Improve the used protocols
- Establish new trust anchors
- Secure Channels
- Develop new standards
- Roll-out by all browser vendors

=> Improve Security for Applications and the user

# Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

# What's been done / what's coming

- Mime-Sniffing
- Same-Origin Policy
- Secure Channel:
  - HSTS Strict Transport Security
  - TLS in DNSSEC
- Frame-Options
- Content Security Policy
- Do-Not-Track

## Mime-Sniffing

- OS and Browsers use algorithms beyond content-type to identify the application
- Can bypass security protection mechanisms when declared as txt and then later executed as js or pdf
- New standard to unify the way browsers and OS detect content-types

## Same-Origin Policy

- The origin/source of content and code is <u>the</u> important criteria for trust on the Internet
- How to determine whether sources use the same or related origin
- Tuple: scheme/URI/port
- Currently browsers use different methods to identify whether something has the same origin
- Can lead to unintended trust to related but not identical sources spoofing/tampering/unintended

## Secure Channels

Problems:

- establish secure and trusted channels,
- prevent MiM attacks (SSL stripping / SSL downgrading)

Approaches:

- Strict Transport Security
- TLS in DNSSEC

## Secure Channels: Strict Transport Security

- Server declares “I only talk TLS”
- Example:
  HTTP(S) Response Header:
  Strict-Transport-Security: max-age=15768000; includeSubDomains
- Header can be cached and also prevents leakage via subdomain-content through non-TLS links in content
- Weakness: “Trust on first use”

# Secure Channels: DNSSEC for TLS

- DNSSEC can be used to declare supported protocols for domains
- DNSSEC can be used to declare server certificate for domain

- Advantage: Advantage of trusted signed source
- Disadvantage: long time to deploy

## Frame-Options – Example Use-Cases

A.1. Shop

- An Internet Marketplace/Shop link/button to "Buy this" Gadget, wants their affiliates to be able to stick the "Buy such-and-such from XYZ" IFRAMES into their pages.

A.2. Confirm Purchase Page

- Onlineshop "Confirm purchase" anti-CSRF page. The Confirm Purchase page must be shown to the end user without possibility of overlay or misuse by an attacker.

## Frame-Options - History

X-Frame-Options

- HTTP-Header:
  - DENY: cannot be displayed in a frame, regardless of the site attempting to do so.
  - SAMEORIGIN: can only be displayed if the top-frame is of the same “origin” as the page itself.

# Frame-Options - draft

Frame-Options: In EBNF: Frame-Options = "Frame-Options" ":" "DENY"/ "SAMEORIGIN" / ("ALLOW-FROM" ":" Origin-List)

- DENY: The page cannot be displayed in a frame, regardless of the site attempting to do so.
- SAMEORIGIN: can only be displayed in a frame on the same origin as the page itself.
- ALLOW-FROM: can only be displayed in a frame on the specified origin(s)

# Content Security Policy

HTTP-Header: content-security-policy = "X-Content-Security-Policy:" OWS [ policy ] OWS

Directives (1)

- default-src:
- script-src: <script> elements
- object-src: <object>, <embed> and <applet> elements.
- img-src: <img> elements, CSS properties and shortcut icons, or favicons
- media-src: <video> elements and <audio> elements

## Content Security Policy

Directives (2)

- style-src: <link rel="stylesheet"> elements, or external stylesheets
- frame-src: sources from where permitted to load <iframe> elements
- font-src: load fonts using the @font-face CSS rule
- xhr-src: connected to via XMLHttpRequest objects
- (frame-ancestors: permitted to embed the protected resource as an <iframe>, <frame> or <object> element)
- report-uri: URIs to which a violation report is sent when a policy violation occurs
- policy-uri: (location of a file containing the policy)
- Options:

## Not security - but related: Privacy

- Do-Not-Track
- HTTP-Request-Header to indicate that a user does not want a web server to use advertising-tracking to track his behaviour/identity
- To be enforced through legal and regulatory policy on the server side

## Browser Security

- History
- What’s the problem
- Who & Why
- What’s been done
- When

## When - Timeframes

- Mime-Sniffing - Q3/2011
- Same-Origin – Q4/2011
- HSTS Strict Transport Security – Q4/2011
- Frame-Options – Q4/2011
- Content Security Policy - 2012
- TLS in DNSSEC - 2012
- Do-Not-Track – 2012+

# Join the discussion

Ideas / feedback / participation welcome

IETF Websec:
http://tools.ietf.org/wg/websec/charters

W3C Web App Sec:
http://www.w3.org/2010/07/appsecwg-charter

Or drop me an email:
tobias.gondrom@gondrom.org

???

Questions?

Thank you