Time does not heal all wounds: A longitudinal analysis of security-mechanism support in mobile browsers

Meng Luo, Pierre Laperdrix, Nima Honarmand, Nick Nikiforakis

NDSS Symposium
San Diego, February 25th, 2019
Traffic from mobile vs. desktop

- Mobile: 52.2%
- Desktop: 47.8%
Client-side attacks/preventions

**ATTACKS**

Cross-site scripting (XSS), cross-site request forgery (CSRF), SSL stripping, clickjacking, ...

**DEFENSES**

**Isolating different origins:**
Same-origin policy (SOP), Content Security Policy (CSP), X-Frame-Options, iframe sandboxing

**Defending against man-in-the-middle attacks:**
HTTP Strict-Transport-Security (HSTS), block-all-mixed-content, upgrade-insecure-requests

**Protecting cookies and enhancing privacy:**
Security-related flags for HTTP cookies, referrer policy
Motivation

• Hundreds of mobile browsers are available in the market – each advertising unique features
  – Built-in anti-tracking capabilities
  – Voice control
  – Increased performance
• Two users may get substantially different security guarantees depending on the browser they utilize
• No prior work on the adoption of security mechanisms in mobile browsers
Automated testing framework

Hindsight: Understanding the Evolution of UI Vulnerabilities in Mobile Browsers

Meng Luo, Oleksii Starov, Nima Honarmand, Nick Nikiforakis
{melo, ostarov, nhonarmand, nick}@cs.stonybrook.edu

ABSTRACT

Much of recent research on mobile security has focused on malicious applications. Although mobile devices have powerful browsers
commonly used by users and are vulnerable to at least as many
topogaphs, work emails, and financial information for attacks.

Even though the most common form of abuse in smart devices
is that of malicious applications, it is most certainly not
possible kind of abuse. One must not forget that smartphones
are susceptible to at least as many problems as desktop browsers. A user visiting a malicious
through her mobile browser can be the victim of web
attacks (e.g., XSS and CSRF), attacks against the browser
(24) and application logic issues
among

Top 20 mobile browsers
# Security mechanisms

<table>
<thead>
<tr>
<th>Category</th>
<th>Content</th>
<th># tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same-origin Policy</td>
<td>DOM access, cookie scope, XMLHttpRequest and worker</td>
<td>33</td>
</tr>
<tr>
<td>Content Security Policy</td>
<td>Fetch (e.g. script-src) and other directives (e.g. form-action, frame-ancestors and upgrade-insecure-requests)</td>
<td>253</td>
</tr>
<tr>
<td>Cookie</td>
<td>Secure, HttpOnly and SameSite flags</td>
<td>11</td>
</tr>
<tr>
<td>Referrer policy</td>
<td>no-referrer-when-downgrade (default) and other values</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>(e.g. no-referrer, origin, same-origin and strict-origin)</td>
<td></td>
</tr>
<tr>
<td>Iframe sandbox</td>
<td>JavaScript execution, form submission and top-level navigation</td>
<td>3</td>
</tr>
<tr>
<td>X-Frame-Options</td>
<td>Deny, SameOrigin and Allow-From values</td>
<td>30</td>
</tr>
<tr>
<td>Strict-Transport-Security</td>
<td>Basic and includeSubDomains value</td>
<td>2</td>
</tr>
<tr>
<td>X-Content-Type-Options</td>
<td>Script sniffing opt-out</td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>395</td>
</tr>
</tbody>
</table>
Evaluation

• 395 tests × 351 browsers → >138K vulnerability reports

• Gauging the success/failure of a test,
  – Support -> “secure”
  – Lack of support -> “vulnerable”

• Analyzing vulnerability reports:
  – Longitudinal analysis capturing evolution of security-mechanism support
  – Dependencies between security-mechanism support and underlying Android system
  – Case studies of common vulnerabilities
Longitudinal analysis
Adoption trend (1)

• Do browsers support more security mechanisms over time?

UC vs. Firefox: more downloads, more vulnerabilities

Jiubang & Boat browser: Vulnerable to >60 tests (millions of downloads), not updated since 2016

CSP is excluded
Adoption trend (2)

- Is there any difference in the support rate of security mechanisms?

**HSTS** starts with little support, but is eventually as widely supported as Same-origin Policy.
Window of vulnerability (1)

• Time window of vulnerability

• Crawl snapshots of Alexa top 5K websites
• Obtain the earliest time when any website first utilizes a given security mechanism
# Window of vulnerability (2)

- The adoption of security mechanisms on mobile browsers is significantly slower than on desktop browsers

<table>
<thead>
<tr>
<th>Security mechanisms</th>
<th>Website request</th>
<th>Chrome desktop</th>
<th>Firefox desktop</th>
<th>First mobile support</th>
<th>50% mobile support</th>
<th>75% mobile support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSP</td>
<td>2011</td>
<td>2011</td>
<td>2011</td>
<td>2011</td>
<td>2014 (+3)</td>
<td>2015 (+4)</td>
</tr>
<tr>
<td>Cookie</td>
<td>&lt;2011</td>
<td>2011</td>
<td>2009</td>
<td>2011</td>
<td>2013 (+2)</td>
<td>2014 (+3)</td>
</tr>
<tr>
<td>Referrer-Policy</td>
<td>2016</td>
<td>2012</td>
<td>2015</td>
<td>2015</td>
<td>2018 (+3)</td>
<td>Not yet</td>
</tr>
<tr>
<td>X-Frame-Options</td>
<td>&lt;2011</td>
<td>2010</td>
<td>2010</td>
<td>2011</td>
<td>2013 (+2)</td>
<td>2014 (+3)</td>
</tr>
<tr>
<td>X-Content-Type-Options</td>
<td>&lt;2011</td>
<td>2008</td>
<td>2011</td>
<td>2011</td>
<td>2013 (+2)</td>
<td>2015 (+4)</td>
</tr>
</tbody>
</table>
Security regression (1)

• Definition: a browser version stops supporting a security mechanism that was supported by an earlier version

• Types of security regressions:
  – temporary regression
  – permanent regression

• 55% browser families show regressions
Security regression (2)

Given the complexity of Content Security Policy (CSP), most regressions occur in adopting CSP.
Security regression (3)

- Security regressions in top-5 browsers
- The red symbol indicates the presence of security regression

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Chrome</th>
<th>UC</th>
<th>Firefox</th>
<th>Opera</th>
<th>Opera mini</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CSP</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cookie</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Referrer-Policy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>X-Frame-Options</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HSTS</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>X-Content-Type-Options</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sandbox</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

UC browser shows frequent security regressions

Opera stops supporting HSTS from v.15 to v.26 (1 year), thereby exposing users to MITM attacks
Studying latest mobile browsers
Relationship between security and underlying OS

• Many mobile browsers are developed using WebView
• The version of WebView depends on the underlying Android system
  — More recent versions support more security mechanisms
• Two users who utilize the latest version of the same browser, can experience vastly different levels of security.
Vulnerability vs. Android version

The security level is promoted (less vulnerabilities), as newer Android system versions are used.
A common vulnerability is: “Allow-From” value of X-Frame-Options header is not respected
Vulnerability vs. Android version

UC Mini browser has the exact same and biggest number of vulnerabilities regardless of Android versions.
Case studies: the impact of browser vulnerabilities on web apps
Anti-clickjacking

- X-Frame-Options is an anti-clickjacking mechanism
- Chrome and WebView-based browsers discard the entire header when the “allow-from” directive is used
- 231/10,752 websites (from Alexa top 50K) that make use of the X-Frame-Options, are using “allow-from”
- 175/231 websites do not utilize CSP’s “frame-ancestors”
  - Most of these websites have user accounts that could be abused through a clickjacking attack
  - American and Russian banking sites
  - Government sites of US, China, Brazil and India
  - Cloud instrumentation services
CSRF and SameSite

- CSRF occurs when a malicious website forges requests to perform unwanted actions on a vulnerable web app on behalf of an authenticated user.
- The SameSite mechanism prevents browsers from attaching cookies to cross-site requests.
- 93 websites (from Alexa top 50K) utilize SameSite, including an Italian bank and the biggest streaming platform.
- Users of the above web apps may not be protected against CSRF if they use mobile browsers that do not support this mechanism—UC browser, Opera Mini.
Conclusion

• Developed 395 tests to evaluate security-mechanism support

• 395 tests × 351 mobile browsers → 138K tests

• Analyzed the evolution of security-mechanism support
  – Quantified trends in the support of different security mechanisms
  – Multi-year window of vulnerability for all security mechanisms
  – Observed security regressions

• Security-mechanism support may depend on the underlying OS

• We need tools that can adapt to a user’s mobile environment and employ different security mechanisms
  – E.g., upon detecting that ‘allow-from’ is not supported, emit ‘frame-ancestors’ CSP directive
THANK YOU

meluo@cs.stonybrook.edu