Cookie Crumbles: Breaking and Fixing Web Session Integrity

Marco Squarcina (TU Wien)
32nd USENIX Security Symposium
August 9-11, 2023

Joint work with
Pedro Adão¹
Lorenzo Veronese²
Matteo Maffei²

¹ IST, Universidade de Lisboa, ² TU Wien

@blueminimal
https://infosec.exchange/@minimalblue
marco.squarcina@tuwien.ac.at
M. Squarcina // Cookie Crumbles: Breaking and Fixing Web Session Integrity
Have Weak Integrity
8.6. Weak Integrity

Cookies do not provide integrity guarantees for sibling domains (and their subdomains). For example, consider foo.site.example and bar.site.example. The foo.site.example server can set a cookie with a Domain attribute of "site.example" (possibly overwriting an existing "site.example" cookie set by bar.site.example), and the user agent will include that cookie in HTTP requests to bar.site.example. In the worst case, bar.site.example will be unable to distinguish this cookie from a cookie it set itself. The foo.site.example server might be able to leverage this ability to mount an attack against bar.site.example. [...] 

An active network attacker can also inject cookies into the Cookie header field sent to https://site.example/ by impersonating a response from http://site.example/ and injecting a Set-Cookie header field. The HTTPS server at site.example will be unable to distinguish these cookies from cookies that it set itself in an HTTPS response. An active network attacker might be able to leverage this ability to mount an attack against site.example even if site.example uses HTTPS exclusively. [...] 

Finally, an attacker might be able to force the user agent to delete cookies by storing a large number of cookies. Once the user agent reaches its storage limit, the user agent will be forced to evict some cookies. Servers SHOULD NOT rely upon user agents retaining cookies.

Typical attacks: Session Fixation, Login CSRF, CSRF, application specific vulns, ...
8.6. Weak Integrity

Cookies do not provide integrity guarantees for sibling domains (and their subdomains).

Domain attackers will include

"site.example.com "cookie Quack"

be able to bar.site.example.com

An active header field response

An active mount and

Finally, cookies

Typical attacks: Session Fixation, Login CSRF, CSRF, application specific vulns, ...
8.6. Weak Integrity

Cookies do not provide integrity guarantees for their subjects.

Domain and "site.ex" will interact. The worst case is
cookie timing. We be able to bar.site

An active header for response field. The cookies can be

An active amount and exclusive

Finally, cookies

reaches a some cookies.

Typical attacks: Session Fixation, Login CSRF, CSRF, application specific vulns, ...
Recap: **Cookie Tossing (Same-Site & Network Attackers)**

Cookies do not follow the Same Origin Policy.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expires</td>
<td>Domain</td>
</tr>
<tr>
<td>Max-Age</td>
<td>Path</td>
</tr>
<tr>
<td></td>
<td>SameSite</td>
</tr>
<tr>
<td></td>
<td>Secure</td>
</tr>
<tr>
<td></td>
<td>HttpOnly</td>
</tr>
</tbody>
</table>

Path useful to prioritize cookies.
Cross-Origin Request Forgery (CORF) Protections

- **SameSite** attribute does not apply in same-site context → **Token-based defenses**!
- **Double-Submit Pattern** (DSP) is broken!

POST /action

Cookie: session=SGkgVVNFTk1YIQ==; csrf=X
- csrf_tok=X

No integrity
Cross-Origin Request Forgery (CORF) Protections

- **SameSite** attribute does not apply in same-site context → **Token-based defenses**!
- **Double-Submit Pattern (DSP)** is broken!

**POST /action**

Cookie: session=SGkgVVNFTk1YIQ==; csrf=✓
  - csrf_tok=✓

**No integrity**

- **Synchronizer Token Pattern (STP)**

**POST /action**

Cookie: session={csrf_secret:VVNFTk1Y, id:Marco}
  - csrf_tok=VVNFTk1Y

**Validation**

csrf_tok == fun(csrf_secret, args)
Cross-Origin Request Forgery (CORF) Protections

- **SameSite** attribute does not apply in same-site context → **Token-based defenses**!
- **Double-Submit Pattern (DSP)** is broken!

![Diagram showing CORF protections with examples of POST /action requests and validation of csrf_tok]

- **Synchronizer Token Pattern (STP)**
CORS Token Fixation (Flask-login + Flask-WTF)

1. Attacker obtains CSRF token $t_0$ and $sess_0$
   
   $sess_0 = \{\text{csrf:s, id:None}\}$

2. GET /login
   
   Pre-Login Session Fixation with session=$sess_0$
CORG Token Fixation (Flask-login + Flask-WTF)

1. Attacker obtains CSRF token \( t_0 \) and \( sess_0 \)
   \[ sess_0 = \{ \text{csrf:s, id:None} \} \]

2. GET /login
   Pre-Login Session Fixation with \( session=sess_0 \)

3. User authenticates with \( sess_0 \), obtains \( sess_1 \)
   \[ sess_1 = \{ \text{csrf:s, id:bob} \} \]

4. POST /action
   Cookie: session={csrf:s, id:bob}
   - csrf_token=t0

CSRF token \( t_0 \) is valid!
## Web Frameworks Analysis

<table>
<thead>
<tr>
<th>Framework</th>
<th>Broken STP</th>
<th>Default DSP</th>
<th>Session Fixation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Express (passport + csurf)</td>
<td>●</td>
<td></td>
<td>●</td>
<td><strong>CVE-2022-25896</strong></td>
</tr>
<tr>
<td>Koa (koa-passport + csrf)</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fastify (fastify/passport + csrf-protection)</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td><strong>CVE-2023-29020</strong> <strong>CVE-2023-27495</strong> <strong>CVE-2023-29019</strong></td>
</tr>
<tr>
<td>Sails* (csrf)</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flask (flask-login+flask-wtf)</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tornado</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symfony (security-bundle)</td>
<td>●</td>
<td></td>
<td></td>
<td><strong>CVE-2022-24895</strong></td>
</tr>
<tr>
<td>CodeIgniter4 (shield)</td>
<td>●</td>
<td>●</td>
<td></td>
<td><strong>CVE-2022-35943</strong></td>
</tr>
<tr>
<td>Yii2</td>
<td></td>
<td>●</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* affects the bootstrap template app
Are Cookie Crumbles Getting Better?
Methodology

1. Manual review of the cookie standard
2. Browser testing suite for tossing & eviction: check discrepancies
3. Simple differential fuzzing to test server implementations

rfc6265bis
Strict Secure & Cookie Prefixes [__Host-]

HTTP Working Group
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: March 9, 2017

Deprecate modification of 'secure' cookies from non-secure origins
draft-ietf-httpbis-cookie-alone-01

HTTP Working Group
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: August 26, 2016

Cookie Prefixes
draft-ietf-httpbis-cookie-prefixes-00

block setting cookie without the Secure flag if the cookie jar contains Secure cookie with the same name
Strict Secure & Cookie Prefixes (__Host-)

HTTP Working Group
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: March 9, 2017

Deprecate modification of 'secure' cookies from non-secure origins
draft-ietf-httpbis-cookie-alone-01

HTTP Working Group
Internet-Draft
Updates: 6265 (if approved)
Intended status: Standards Track
Expires: August 26, 2016

>`document.cookie = '__Host-sess=bar; Path=/; Secure; Domain=example.com'
`'

High-integrity cookies, cannot be set from a sibling domain!

block setting cookie without the Secure flag if the cookie jar contains Secure cookie with the same name
## Browser vs Server

<table>
<thead>
<tr>
<th>Set-Cookie:</th>
<th>Cookie:</th>
<th>Key</th>
<th>Value</th>
<th>Server &lt;key, value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo=</td>
<td>foo=</td>
<td>foo</td>
<td></td>
<td>&lt;foo, &gt;</td>
</tr>
<tr>
<td>=foo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=foo=</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>==foo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>foo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Browser vs Server

<table>
<thead>
<tr>
<th>Set-Cookie:</th>
<th>Cookie:</th>
<th>Key</th>
<th>Value</th>
<th>Server &lt;key, value&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo=</td>
<td>foo=</td>
<td>foo</td>
<td></td>
<td>&lt;foo, &gt;</td>
</tr>
<tr>
<td>=foo</td>
<td>foo</td>
<td>foo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>=foo=</td>
<td>foo=</td>
<td>foo=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>==foo</td>
<td>=foo</td>
<td>=foo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>foo</td>
<td>foo</td>
<td>foo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Browser vs Server

<table>
<thead>
<tr>
<th>Set-Cookie:</th>
<th>Cookie:</th>
<th>Key</th>
<th>Value</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo=</td>
<td>foo=</td>
<td>foo</td>
<td></td>
<td>&lt;foo,</td>
</tr>
<tr>
<td>=foo</td>
<td>foo</td>
<td></td>
<td>foo</td>
<td>&lt;foo,</td>
</tr>
<tr>
<td>=foo=</td>
<td>foo=</td>
<td></td>
<td>foo=</td>
<td>&lt;foo,</td>
</tr>
<tr>
<td>==foo</td>
<td>=foo</td>
<td></td>
<td>=foo</td>
<td>&lt;foo,</td>
</tr>
<tr>
<td>foo</td>
<td>foo</td>
<td></td>
<td>foo</td>
<td>&lt;foo,</td>
</tr>
</tbody>
</table>

CVE-2023-23934

Werkzeug <2.2.3

M. Squarcina // Cookie Crumbles: Breaking and Fixing Web Session Integrity
### Browser vs Server

**[RFC6265bis] Accept nameless cookies. (#1018)**

This patch alters the cookie parsing algorithm to treat `Set-Cookie: token` as creating a cookie with an empty name and a value of "token". It also rejects cookies with neither names nor values (e.g. `Set-Cookie:` and `Set-Cookie: =`).

Closes #159.

`main` (#1018)

`draft-ietf-httpbis-unprompted-auth-02` ...

![commit](b68e4ff)

committed on Jan 10, 2020

<table>
<thead>
<tr>
<th>Set</th>
<th>Value</th>
<th>Server</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo</td>
<td>foo</td>
<td>&lt;foo,</td>
<td>&gt;</td>
</tr>
<tr>
<td>=fo</td>
<td>foo</td>
<td>&lt;foo,</td>
<td>&gt;</td>
</tr>
<tr>
<td>=fo</td>
<td>foo</td>
<td>&lt;foo,</td>
<td>&gt;</td>
</tr>
<tr>
<td>==fo</td>
<td>foo</td>
<td>&lt;foo,</td>
<td>&gt;</td>
</tr>
<tr>
<td>foo</td>
<td>foo</td>
<td>&lt;foo,</td>
<td>&gt;</td>
</tr>
</tbody>
</table>
Real World Implications

- **Bypass __Host-cookies**

```plaintext
Set-Cookie: =__Host-sess=bad; domain=bank.com
Cookie: __Host-sess=bad
```

Value of a nameless cookie __Host-sess=bad

* Reported almost simultaneously with Axel Chong, our issues were merged to jointly discuss mitigations and additional security implications. See also https://github.com/httpwg/http-extensions/issues/2229

CVE-2022-2860*  CVE-2022-40958*

Fixed in rfc6265bis and browsers
Real World Implications

- **Bypass __Host-cookies**
  
  Set-Cookie: =__Host-sess=bad; domain=bank.com
  
  Cookie: __Host-sess=bad

- **Bypass Strict Secure cookies**
  
  Set-Cookie: sess=good; Secure
  
  http://bank.com
  
  Set-Cookie: =sess=bad; Path=/app/
  
  http://bank.com
  
  Cookie: sess=bad; sess=good
  
  https://bank.com/app/

* Reported almost simultaneously with Axel Chong, our issues were merged to jointly discuss mitigations and additional security implications. See also https://github.com/httpwg/http-extensions/issues/2229

CVE-2022-2860*  CVE-2022-40958*
• 12 CVEs, 27 vulnerability reports

• More browser issues, client-server discrepancies, server parsing vulnerabilities

• Cookie measurement for prefixes and nameless cookies

• Formal modeling of (patched) Web frameworks using ProVerif

Artifact

https://github.com/SecPriv/cookiecrumbles
... and that's the way the cookie crumbles!
Thank You! Questions?

Marco Squarcina (TU Wien)

@blueminimal
https://infosec.exchange/@minimalblue
marco.squarcina@tuwien.ac.at