

Cookie Crumbles:

Breaking and Fixing Web Session Integrity



Marco Squarcina (TU Wien)

32nd USENIX Security Symposium

August 9-11, 2023



@blueminimal



<https://infosec.exchange/@minimalblue>



marco.squarcina@tuwien.ac.at



Pedro Adão¹



Lorenzo Veronese²



Matteo Maffei²

Joint work with

¹ IST, Universidade de Lisboa, ² TU Wien





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.

RFC 6265bis

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). For example, if a cookie is set for bar.site, it will be sent to any site that shares the same domain as bar.site. This is a security issue because an active attacker can use this to inject malicious code into the cookie. The worst case is that the cookie will be able to be used by bar.site to access sensitive information.

An active attacker can use this to inject malicious code into the cookie. The worst case is that the cookie will be able to be used by bar.site to access sensitive information. An active attacker can use this to inject malicious code into the cookie. The worst case is that the cookie will be able to be used by bar.site to access sensitive information.

Finally, cookies are not secure. They can be intercepted by an attacker on the network. This is a security issue because an active attacker can use this to inject malicious code into the cookie. The worst case is that the cookie will be able to be used by bar.site to access sensitive information.



Typical attacks: Session Fixation, Login CSRF, **CSRF**, application specific vulns, ...

8.6. Weak Integrity

Cookies do not provide integrity guarantees for their subdomain. For example, bar.site can set a cookie for bar.site, but bar.site cannot set a cookie for bar.site. This is because the browser will include the cookie for the worst case domain. A cookie for bar.site will be able to be sent to bar.site. An active header for response field. These cookies are active and mount an exclusive. Finally, cookies reach some cookies.



Cookies Lack Integrity: Real-World Implications

Xiaofeng Zheng^{1,2,3}, Jian Jiang⁷, Jinjin Liang^{1,2,3}, Haixin Duan^{1,3,4}, Shuo Chen⁵, Tao Wan⁶, and Nicholas Weaver^{4,7}

¹Institute for Network Science and Cyberspace, Tsinghua University

²Department of Computer Science and Technology, Tsinghua University

³Tsinghua National Laboratory for Information Science and Technology

⁴International Computer Science Institute

⁵Microsoft Research Redmond

⁶Huawei Canada

⁷UC Berkeley



Abstract

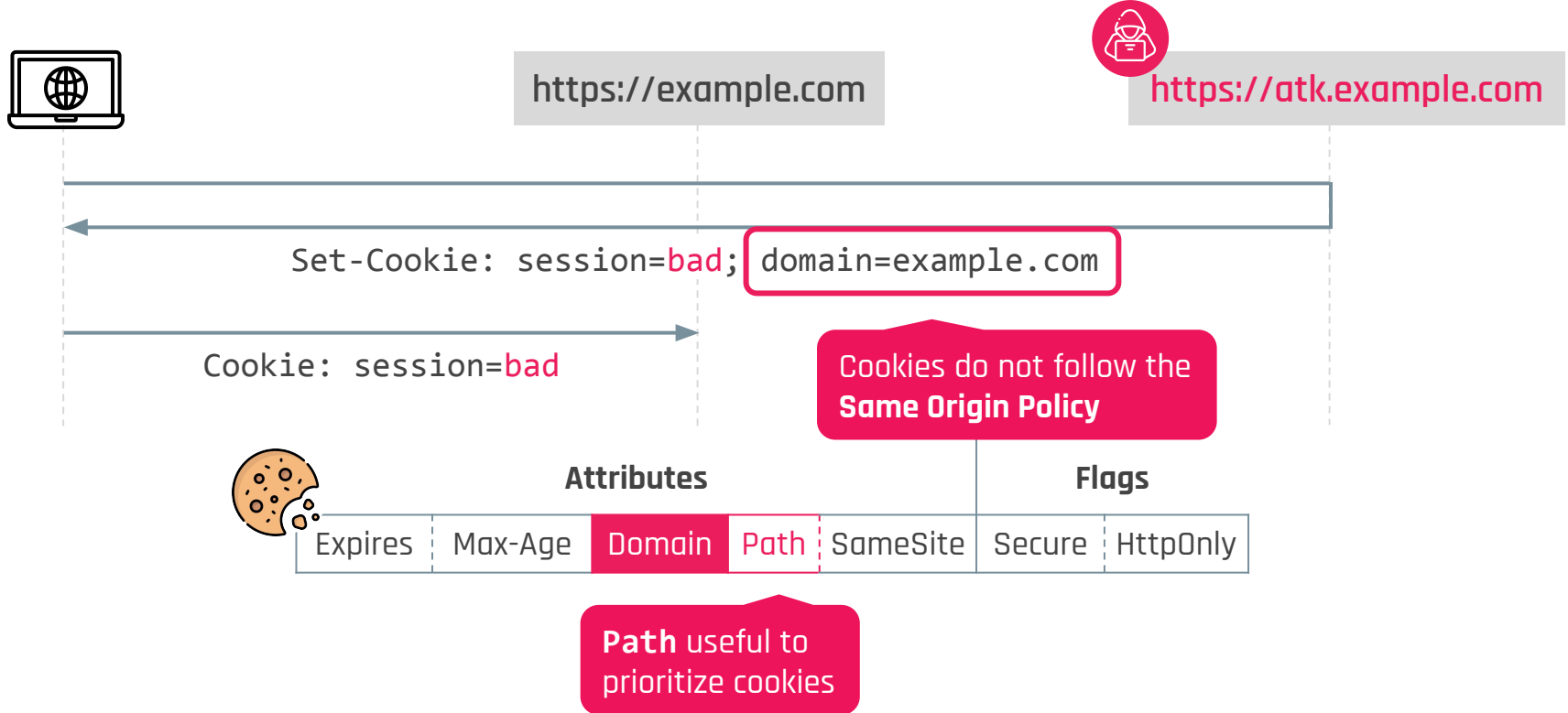
A cookie can contain a “secure” flag, indicating that it should be only sent over an HTTPS connection. Yet there is no corresponding flag to indicate how a cookie was set: attackers who act as a man-in-the-middle even temporarily on an HTTP session can inject cookies which will be attached to subsequent HTTPS connections. Similar attacks can also be launched by a web attacker from a related domain. Although an acknowledged threat, it has not yet been studied thoroughly. This paper aims to fill this gap with an in-depth empirical assessment of cookie injection attacks. We find that cookie-related vulnerabilities are present in important sites (such as Google and Bank of America), and can be made worse by the implementation weaknesses we discovered in major web browsers (such as Chrome, Firefox, and Safari). Our successful attacks have included privacy violation, on-line victimization, and even financial loss and account

man-in-the-middle (MITM). However, there is no similar measure to protect its integrity from the same adversary: an HTTP response is allowed to set a secure cookie for its domain. An adversary controlling a related domain is also capable to disrupt a cookie’s integrity by making use of the shared cookie scope. Even worse, there is an asymmetry between cookie’s read and write operations involving pathing, enabling more subtle form of cookie integrity violation.

The lack of cookie integrity is a known problem, noted in the current specification [2]. However, the real-world implications are under-appreciated. Although the problem has been discussed by several previous researchers [4, 5, 30, 32, 24, 23], none provided in-depth and real-world empirical assessment. Attacks enabled by merely injecting malicious cookies could be elusive, and the consequence could be serious. For example, a cautious user might only visit news websites at open wireless

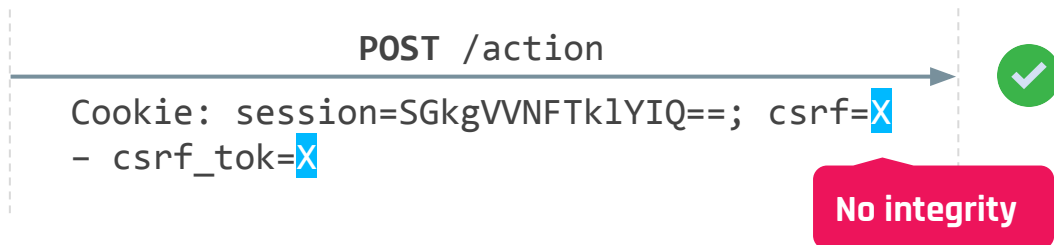
Typical attacks: Session Fixation, Login CSRF, **CSRF**, application specific vulns, ...

Recap: Cookie Tossing (Same-Site & Network Attackers)



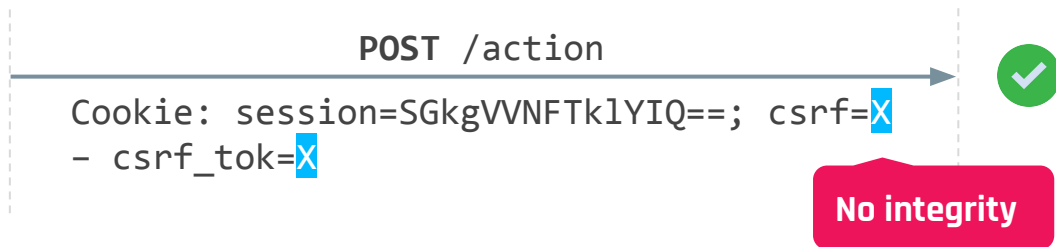
Cross-Origin Request Forgery (CORF) Protections

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

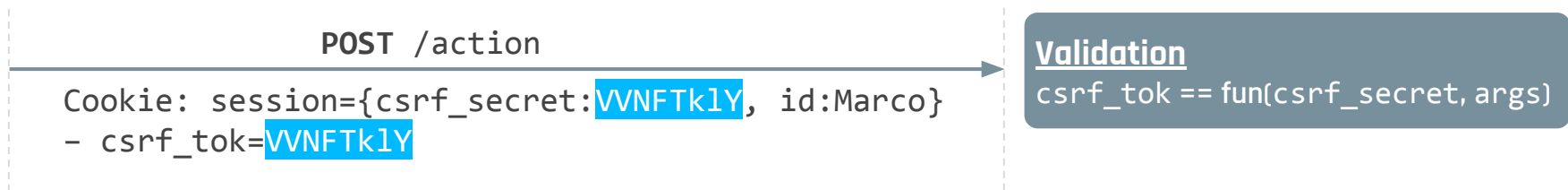


Cross-Origin Request Forgery (CORF) Protections

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

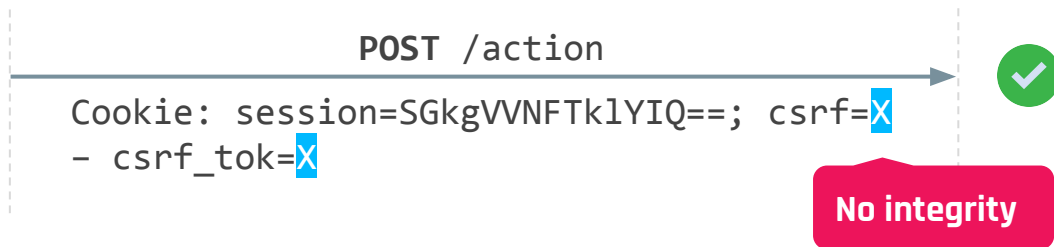


- **Synchronizer Token Pattern (STP)**



Cross-Origin Request Forgery (CORF) Protections

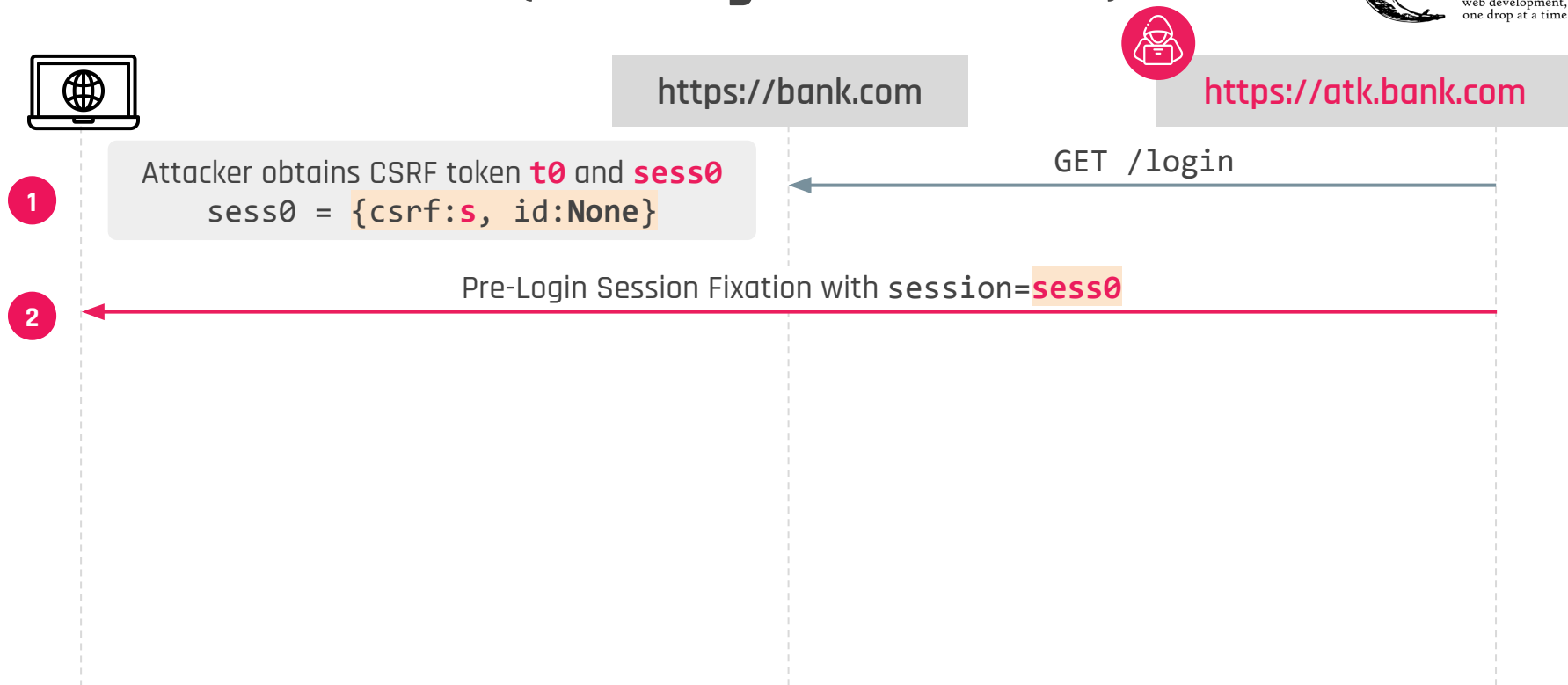
- **SameSite** attribute does not apply in same-site context → Token-based defenses!
- **Double-Submit Pattern (DSP)** is broken!



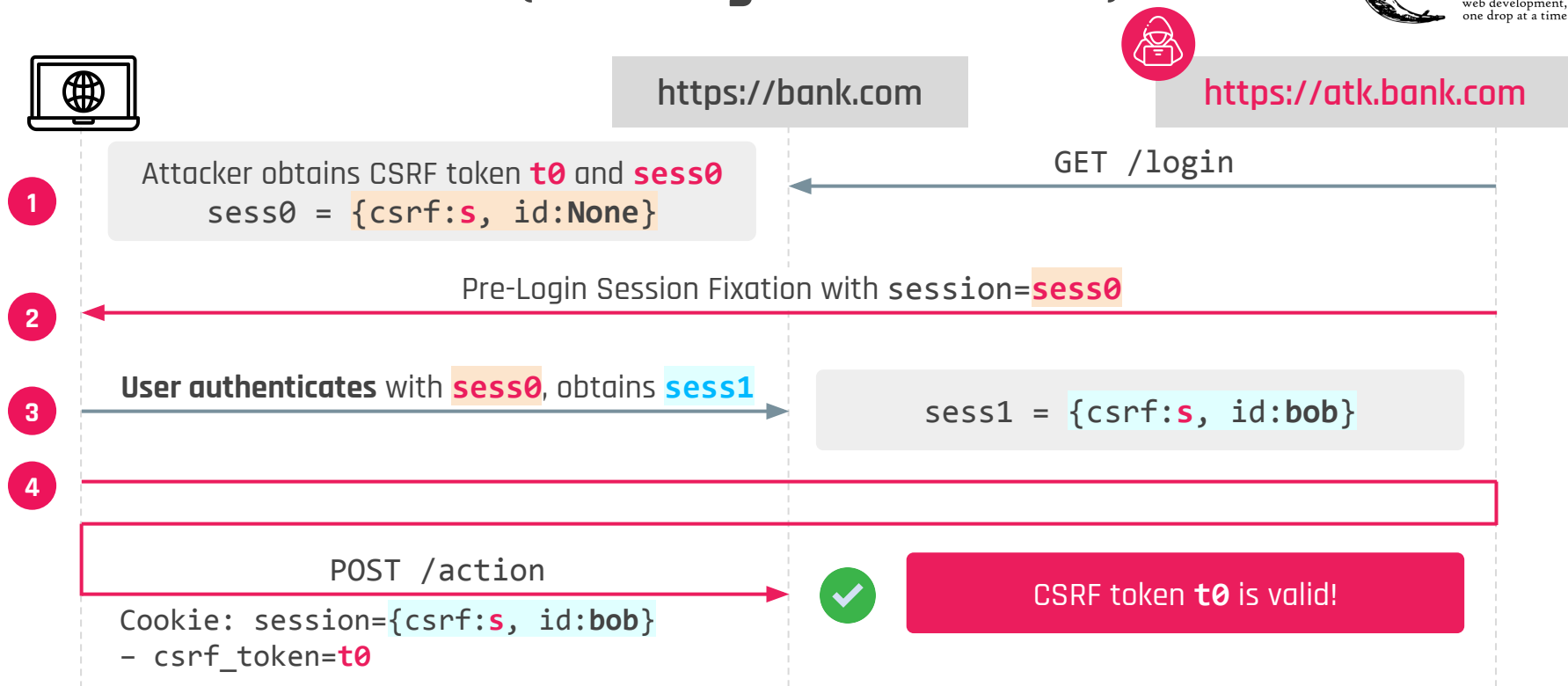
- **Synchronizer Token Pattern (STP)**



CORF Token Fixation (Flask-login + Flask-WTF)



CORF Token Fixation (Flask-login + Flask-WTF)



Web Frameworks Analysis

Framework (9/13 vulnerable)	Broken STP	Default DSP	Session Fixation	
Express (passport + csrf)	●		●	CVE-2022-25896
Koa (koa-passport + csrf)	●			
Fastify (fastify/passport + csrf-protection)	●	●	●	CVE-2023-29020 CVE-2023-27495 CVE-2023-29019
Sails* (csrf)	●		●	
Flask (flask-login+flask-wtf)	●			
Tornado		●		
Symfony (security-bundle)	●			CVE-2022-24895
CodeIgniter4 (shield)	●	●		CVE-2022-35943
Yii2		●		

Homepage
www.passportjs.org/

♥ Fund this package

± Weekly Downloads
 2,042,702

Version
 0.6.0

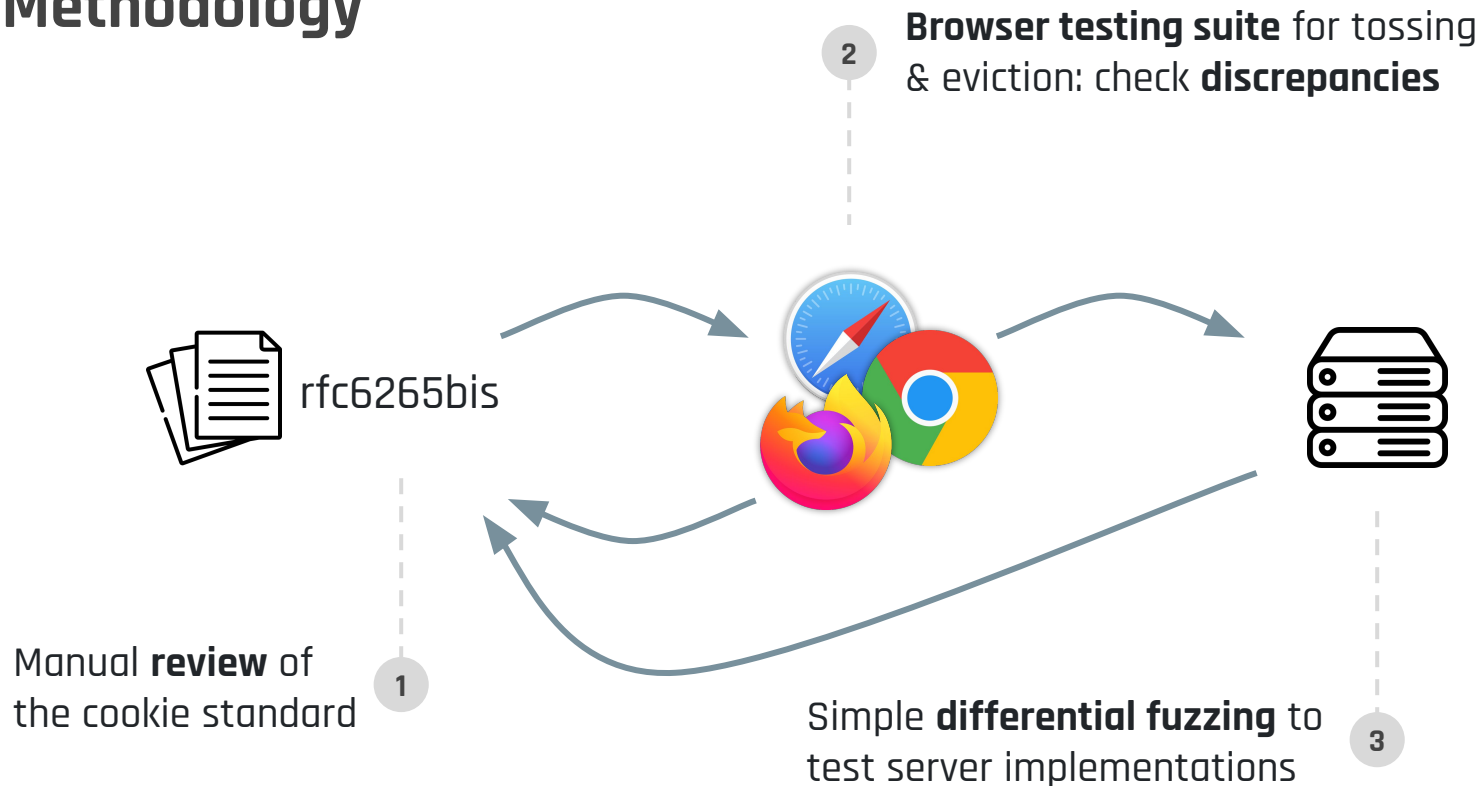
License
 MIT



*affects the bootstrap template app

Are Getting Better?

Methodology



Strict Secure & Prefixes (___Host-)

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

M. West
Google, Inc
September 5, 2016

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

block setting cookie without
the **Secure** flag if the cookie
jar contains Secure cookie
with the same name

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

M. West
Google, Inc
February 23, 2016

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

Strict Secure & Prefixes (__Host-)

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

M. West
Google, Inc
September 5, 2016

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

block setting cookie without the **Secure** flag if the cookie jar contains Secure cookie with the same name

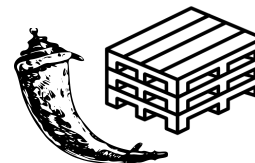
HTTP Working Group
Internet-Draft
Updates: [6265](#) (if approved)
Intended status: Standards
Expires: August 26, 2016

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

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

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

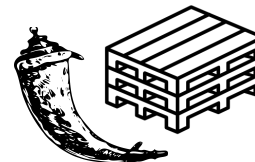
Browser VS Server



Werkzeug <2.2.3

Set-Cookie:	Cookie:	Key	Value	Server <key, value>
foo=	foo=	foo		<foo, >
=foo				
=foo=				
==foo				
foo				

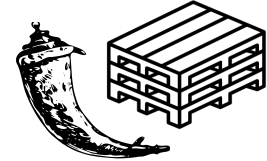
Browser VS Server



Werkzeug <2.2.3

Set-Cookie:	Cookie:	Key	Value	Server <key, value>
foo=	foo=	foo		<foo, >
=foo	foo		foo	
=foo=	foo=		foo=	
==foo	=foo		=foo	
foo	foo		foo	

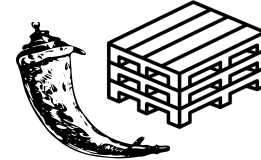
Browser VS Server



Werkzeug <2.2.3

Set-Cookie:	Cookie:	Key	Value	Server <key, value>
foo=	foo=	foo		<foo, >
=foo	foo		foo	<foo, >
=foo=	foo=		foo=	<foo, >
==foo	=foo		=foo	<foo, >
foo	foo		foo	<foo, >

Browser VS Server



Werkzeug <2.2.3

Set

foo

=foo

=foo

==foo

foo

[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: =`).

[Browse files](#)

closes [#159](#).

main (#1018)

draft-ietf-httpbis-unprompted-auth-02 ... **b68e4ff**

committed on Jan 10, 2020

1 parent [c43cdae](#) commit [0178223](#)

Server <key, value>		
	<foo,	>
	<foo,	>
	<foo,	>
	<foo,	>
	<foo,	>

Real World Implications

CVE-2022-2860*

CVE-2022-40958*

- **Bypass __Host- cookies**

Value of a nameless cookie

__Host-sess=bad

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

→ Cookie: __Host-sess=bad

Fixed in
rfc6265bis and
browsers

* 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>

Real World Implications

CVE-2022-2860*

CVE-2022-40958*

- **Bypass __Host- cookies**

Value of a nameless cookie

__Host-sess=bad

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

→ Cookie: __Host-sess=bad

Fixed in
rfc6265bis and
browsers

- **Bypass Strict Secure cookies**

← Set-Cookie: sess=good; Secure

← Set-Cookie: =sess=bad; Path=/app/

→ Cookie: sess=bad; sess=good

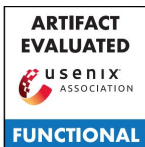
https://bank.com

http://bank.com

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>



- 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>

Cookie Crumbles: Breaking and Fixing Web Session Integrity

Marco Squarcina
TU Wien

Pedro Adão
Instituto Superior Técnico, ULisboa
Instituto de Telecomunicações

Lorenzo Veronese
TU Wien

Matteo Maffei
TU Wien

Abstract

Cookies have a long history of vulnerabilities targeting their confidentiality and integrity. To address these issues, new mechanisms have been proposed and implemented in browsers and server-side applications. Notably, improvements to the Secure attribute and cookie prefixes aim to strengthen cookie integrity against network and same-site attackers, whereas SameSite cookies have been touted as the solution to CSRF. On the server, token-based protections are considered an effective defense for CSRF in the synchronizer token pattern variant. In this paper, we question the effectiveness of these protections and study the real-world security implications of cookie integrity issues, showing how security mechanisms previously considered robust can be bypassed, exposing Web applications to session integrity attacks such as session fixation and cross-origin request forgery (CORF). These flaws are not only implementation-specific bugs but are also caused by compositionality issues of security mechanisms or vulnerabilities in the standard. Our research contributed to 12 CVEs, 27 vulnerability disclosures, and updates to the cookie standard. It comprises (i) a thorough cross-browser evaluation of cookie integrity issues, that results in new attacks originating from implementation or specification inconsistencies, and (ii) a security analysis of the top 13 Web frameworks, exposing session integrity vulnerabilities in 9 of them. We discuss our responsible disclosure and propose practical mitigations.

a session cookie (e.g., via *cross-site scripting*) and use it to obtain unauthorized access to a website [74]. *Session fixation* attacks involve compromising cookie integrity to force an attacker-controlled cookie in the victim's browser, and then impersonate the victim on the target website [63]. *Cross-site request forgery* (CSRF) attacks, instead, are a typical session integrity violation problem where the attacker issues cross-site requests from the victim's browser to execute unwanted actions on a website in which the victim is authenticated [42].

In response to these attacks, new mechanisms have been proposed on both the client and the server side. On the client side, major browsers now support the updated cookie standard RFC6265bis [52] which includes changes compared to the original RFC2965. For example, the *SameSite* attribute is introduced as a robust solution against CSRF attacks. On the server side, changes focused on strengthening the cookie standard include the introduction of the *Secure* flag and the introduction of cookie name prefixes [71]. Client-side protections against CSRF attacks include the use of a token shared between browser and server. This approach has been widely adopted and considered an effective defense against CSRF attacks in the synchronizer token pattern variant [64, 73].

In this paper, we question



... 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

