Who left open the cookie jar?

Tom Van Goethem
OpenEMR security flaws could have exposed millions of patient records

Over 20 severe bugs were found using only manual methods by a single cybersecurity group.

By Charlie Osborne for Zero Day | August 8, 2018 -- 10:40 GMT (11:40 BST) | Topic: Security

The health records of millions of patients worldwide were potentially left open to attack by a slew of critical vulnerabilities uncovered by a single cybersecurity team.

OpenEMR is a popular, open-source software solution for the management of millions of electronic patient records worldwide. However, the software, until recently, also contained over 20 severe security issues.

RECOMMENDED FOR YOU

Guide to Antivirus (AV) Replacement: What You Need to Know Before Replacing Your Current AV Solution

MORE SECURITY NEWS

This is how government spyware StrongPity uses security researchers' work against them

Facebook approaches major cybersecurity firms, acquisition goals in mind

MORE FROM CHARLIE OSBORNE

Security

Most enterprise vulnerabilities...
CSRF Vulnerability in phpMyAdmin 4.7.x Lets Attackers Delete Records through malicious URLs

A Cross-Site Request Forgery (CSRF) vulnerability has been found in the phpMyAdmin version 4.7.x (before version 4.7.7) through which malicious attackers are able to perform fundamental database operations by诱导 users into clicking on maliciously crafted URLs. This vulnerability has been combined under the CVE identifier: CVE-2018-9992. Moreover, this vulnerability is linked to previous CSRF vulnerabilities in phpMyAdmin as
Popular TP-Link wireless home router open to remote hijacking

By concatenating a known improper authentication flaw with a newly discovered CSRF vulnerability, remote unauthenticated attackers can obtain full control over TP-Link TL-WR841N, a popular wireless access point worldwide.
ABSTRACT

Online tracking poses a serious privacy challenge that has drawn significant attention in both academia and industry. Existing approaches for preventing user tracking, based on curated blocklists, suffer from limited coverage and coarse-grained resolution for classification, rely on exceptions that impact sites’ functionality and appearance, and require significant manual maintenance. In this paper we propose a forwards protecting their privacy online. According to [30] ad-blocking usage grew by 70% in 2014, culminating in 41% of people aged between 18-29 using an ad-blocker. This figure is consistent with the results of the empirical evaluation of 200,000 users in Germany presented in this paper.

Any person browsing the Web today is under constant monitoring from entities who track the navigation patterns of users. Previous work [19] reported that 99% of the top 200
Client-side cookie policies

- Defend against the perils of third-party cookies
- Built-in browser options
  - Block third-party cookies
  - Same-site cookies
  - Firefox Tracking Protection
  - Opera Ad Blocker
  - Safari Intelligent Tracking Prevention
- Browser extensions
  - Ad blockers
  - Privacy protection
Client-side cookie policies

- Defend against the perils of third-party cookies

Built-in browser options
- Block third-party cookies
- Same-site
- Firefox Tracking Protection
- Safari Intelligent Tracking Prevention

Browser extensions
- Ad blockers
- Privacy protection

NEEDS TO BE CORRECTLY ENFORCED BY THE BROWSER
Outline

- Background & motivation
- Cookie policy testing framework
- Results
- Evaluating browser security/privacy policies
- Conclusion
Web Fundamentals

Browser

domainA.com/
index.html
domainB.com/
index.html

domainA.com
/index.html
domainB.com
/image.jpg
Web Fundamentals

HTTP cookies

› Implicit inclusion

› Authentication / identification
Web Fundamentals

HTTP cookies

› Implicit inclusion
› Authentication / identification
› Same-Origin Policy

Browser

domainA.com
/index.html

domainB.com
/image.jpg

Domain A

Domain B
Web Fundamentals

HTTP cookies

- Implicit inclusion
- Authentication / identification
- Same-Origin Policy

Browser

Domain A

Domain B
Web Fundamentals

HTTP cookies

- Implicit inclusion
- Authentication / identification
- Same-Origin Policy

![Diagram showing web fundamentals and HTTP cookies](image)
Cross-site request forgery (CSRF)

**Consequence**
Attacker can perform authenticated actions with the victim’s account

**Context**
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, ...)
Cross-site request forgery (CSRF)

Consequence
Attacker can perform authenticated actions with the victim’s account

Context
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, ...)

victim
cute-kittens.com
Cross-site request forgery (CSRF)

Consequence
Attacker can perform authenticated actions with the victim’s account

Context
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, ...)

victim  cute-kittens.com
Cross-site request forgery (CSRF)

**Consequence**
Attacker can perform authenticated actions with the victim’s account

**Context**
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, …)
Cross-site request forgery (CSRF)

Consequence
Attacker can perform authenticated actions with the victim’s account

Context
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, ...)

<img src="https://bank.com/transfer.php?amount=999999&recipient=attacker">
Cross-site request forgery (CSRF)

Consequence
Attacker can perform authenticated actions with the victim’s account

Context
Vulnerabilities detected for big companies in the past (YouTube, banking sites, Netflix, ...)

![Image of victim, cute-kittens.com, and bank.com with an image of a cookie and a transfer button with the amount set to 999999 and recipient set to attacker]
Same-site cookie

- Cookie with additional attribute: SameSite
- Instructed server-side, enforced client-side
  - SameSite=strict $\longrightarrow$ cookie not included in any cross-site requests
  - SameSite=lax $\longrightarrow$ exceptions: top-level GET, prerender
- Google Chrome intends to make SameSite=lax the default
  - Starting from version 76
Use of same-site cookies

bank.com

victim

cute-kittens.com

bank.com
Use of same-site cookies

Set-Cookie: auth=ekSd2lksq090pQDs; SameSite=lax
Use of same-site cookies

Set-Cookie: auth=ekSd2Iksq090pQDs; SameSite=lax
Why evaluate third-party cookie policies?

› Browsers are known to exhibit inconsistent behavior
  ▶ Deviate from standards
  ▶ Unintended side-effects of certain features

› Extensions have been actively bypassed in the past
  ▶ Pornhub exploited WebSockets to circumvent adblockers [1]

Why evaluate third-party cookie policies?

› Browsers are known to exhibit inconsistent behavior
  › Deviate from standards
  › Unintended side-effects of certain features

› Extensions have been actively bypassed in the past
  › Pornhub exploited WebSockets to circumvent adblockers [1]

=> Comprehensive evaluation of effectiveness needed!

Cookie policy testing framework
Framework requirements

› Black box
   ▶ Not all browser source code is available
   ▶ Browsers consist of millions of lines of code

› Needs to support browser extensions

Browser instance
- Browser
- Extension
- Configuration

Framework

Bypass detection data
Framework design
Test-case generation

› Initiate requests using different mechanisms

› Application Cache
  ›› Allows cross-origin caching

› HTML tags
  ›› <script>, <img>, <link>, ...

› Headers
  ›› Link, CSP report, ...

› Redirects

› JavaScript
  ›› Fetch(), EventSource, ...

› PDF JS
  ›› sendForm()

› ServiceWorker API
Tested browser instances

› Browsers
  ›› Chrome
  ›› Opera
  ›› Firefox
  ›› Safari
  ›› Edge
  ›› Tor Browser
  ›› Cliqz

› Extensions
  ›› Ad blocking
    🚫 ABP 🛡️ UD 🟢 🟣
  
  ›› Tracking protection
    🌟 🐻 🦄 🦚 🦫
Results
Results: built-in browser policies

- Blocking third-party cookies
  - Bypassed in Chrome & Opera by JS in PDF (`sendForm()`)  
  - Safari 10 & Edge 40: completely unfunctional

- Built-in tracking protection/ad blocking
  - Opera Ad Blocker & Firefox Tracking Protection: bypasses in several categories
  - E.g. Link: `<http://tracker.com/track/>; rel="prev"`
Results: browser extensions

› All extensions could be bypassed

› Design flaws
  ›› In Chrome, PDFs are rendered in extension + impossible to intercept requests by other requests
  ›› In other browsers: certain requests bypass the extensions

› Unclear API
  ›› No clear distinction for browser background requests

› Common mistakes
  ›› Insufficient permissions to intercept certain requests
PDFium design flaw

GET / POST request

tracker.com
PDFium design flaw

GET / POST request

Extension API

tracker.com
PDFium design flaw

Plugin / Extension

GET / POST request

Extension API

tracker.com
PDFium design flaw

Plugin / Extension

GET / POST request

Extension API

tracker.com
PDFium design flaw

GET / POST request

Extension API
Block third-party cookie option

tracker.com

Plugin / Extension

PDF
JS
Results: same-site cookie

› Chrome & Opera
   ›› SameSite=strict cookie is sent for prerender requests

› Edge
   ›› SameSite=lax bypasses: WebSocket API, <embed>, <object>
   ›› SameSite=strict bypasses: WebSocket API, redirects
Check out our paper: https://tom.vg/papers/who-left-open-the-cookie-jar_usenix18.pdf
Evaluating other browser security/privacy policies
HTTP requests

Browser/system events

disk write

syscall

Test case details
(enforced policy, request method, ...)

Test case

https://test-generator.org/test1
Is **Strict-Transport-Security** correctly implemented in all browsers?

```sql
SELECT * FROM tests WHERE
testcase.header = 'HSTS' AND
request.http_scheme = 'HTTP'
```
Is CSP's `img-src` directive correctly implemented in all browsers?

```sql
SELECT * FROM tests WHERE
testcase.header = 'CSP' AND
  testcase.value CONTAINS 'img-src: none' AND
  request.type = 'image'
```
Browser evaluation framework

- Validate correctness of enforcement of implicit/explicit policies
- Supports all browsers
  - Various configurations
  - Measure influence of browser extensions
- Request triggers can be fuzzed for completeness
- Can be used to validate browser implementation before release
  - New features may introduce side-effects in policies (e.g. prerender)
Conclusion

- Browsers are very complex
  - Many APIs/features, millions LoC

- Extensive evaluation is required
  - Should cover entire “ecosystem”: different request mechanisms, browser extensions, ...

- Several issues discovered for cookie policies
  - Bypasses for all browser extensions + several built-in browser policies

- Framework for evaluating browser security/privacy policies
Questions?

https://WhoLeftOpenTheCookieJar.eu

@tomvangoethem
@GJFR_