OWASP
Open Web Application Security Project

Cookie Security

Myths and Misconceptions

David Johansson – OWASP London 30 Nov. 2017

About Me

- David Johansson (@securitybits)
  - Security consultant with 10 years in AppSec
  - Helping clients design and build secure software
  - Develop and deliver security training
  - Based in London, working for Synopsys

Cookie Security

- Why talk about Cookie Security?

Cookie security is somewhat broken…

Agenda

- Cookie Basics
- The ‘Secure’ Attribute
- The ‘HttpOnly’ Attribute
- The ‘Path’ Attribute
- The ‘Domain’ Attribute
- Cookie Lifetime
- Modern Cookie Protections
- Summary

Background

COOKIE BASICS

History of HTTP Cookies

Cookies are based on an old recipe:

- 1994 –Netscape draft
- 1997 – RFC 2109
- 2000 – RFC 2965
- 2002 – HttpOnly
- 2011 – RFC 6265
- 2017 – RFC 6265bis (draft)

"Classic Film" (https://www.flickr.com/photos/29069717@N02/)

HTTP Cookies

- Cookies are sent in HTTP headers

Server response

```
HTTP/1.1 200 OK
…
Set-Cookie:
id=2bf353246gf3; Secure;
HttpOnly
Set-Cookie: lang=en;
Expires=Wed, 09 Jun 2021
10:18:14 GMT
```

Subsequent client request

```
GET /index.html HTTP/1.1
…
Cookie: id=2bf353246gf3;
lang=en
```

- Attributes influence how cookies are managed by the client (e.g., browser)

Keeping Cookies Secure from Network-level Attackers

THE ‘SECURE’ ATTRIBUTE

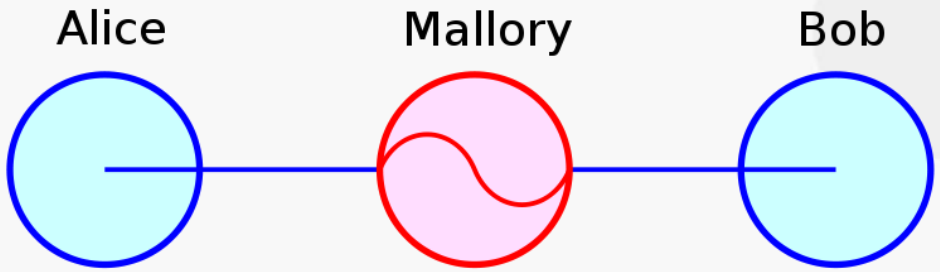
The ‘Secure’ Attribute

“Cookies marked with the ‘Secure’ attribute are only sent over encrypted HTTPS connections and are therefore safe from man-in-the-middle attacks.”

– True or false?

The ‘Secure’ Attribute

- The ‘Secure’ attribute only protects the **confidentiality** of a cookie against MiTM attackers – there is no integrity protection!*



  - Mallory can’t read ‘secure’ cookies
  - Mallory can still **write/change** ‘secure’ cookies

Keeping JavaScript's Hands Away from the Cookie Jar

THE 'HTTPONLY' ATTRIBUTE

CONNECT. LEARN. GROW.

The ‘HttpOnly’ Attribute

“Cookies marked with the ‘HttpOnly’ attribute are not accessible from JavaScript and therefore unaffected by cross-site scripting (XSS) attacks.”

– **True or false?**

The ‘HttpOnly’ Attribute

- Only confidentiality protected in practice
- HttpOnly-cookies can be replaced by overflowing the cookie jar from JavaScript

Picture by Greg Putrich (flickr.com)

Overwriting a Cookie Marked as ‘HttpOnly’ from JavaScript

DEMO

Isolating Cookies to Specific Paths

THE ‘PATH’ ATTRIBUTE

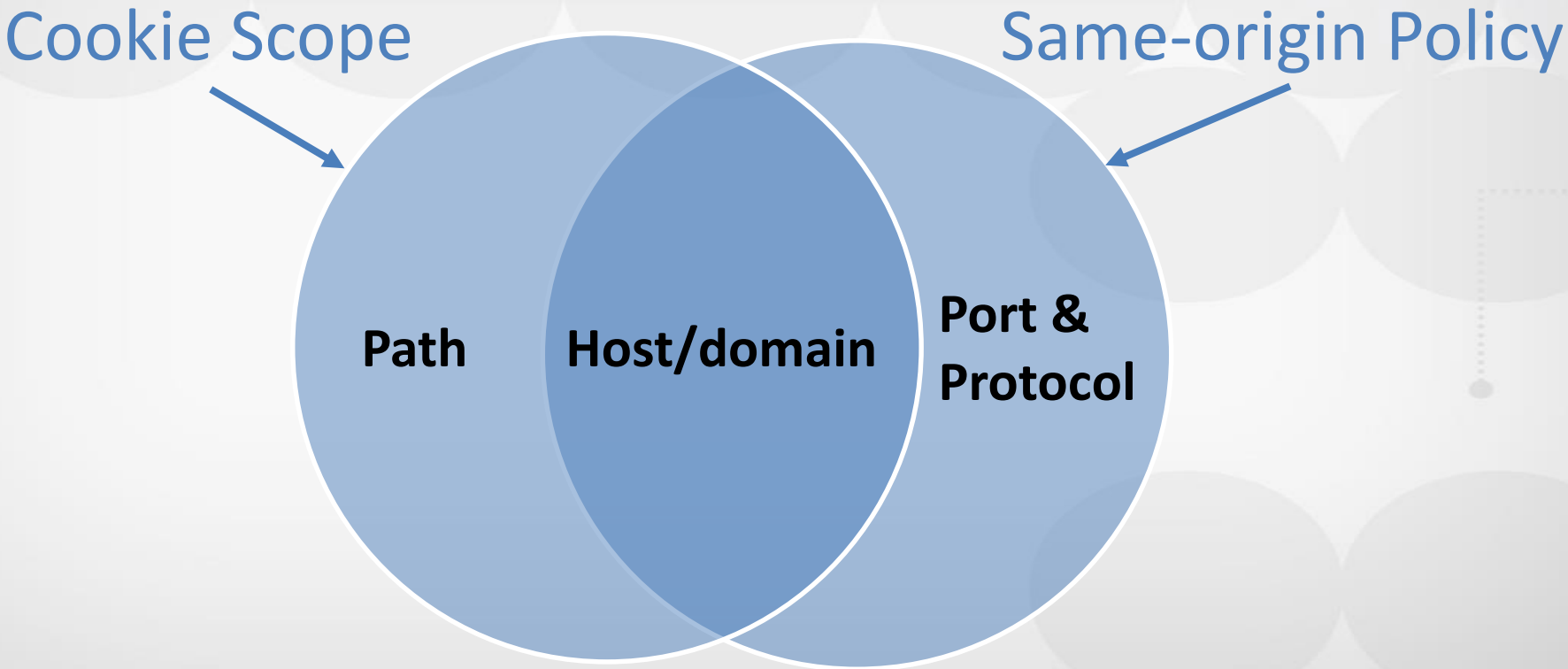
The ‘Path’ Attribute

“The ‘Path’ attribute limits the scope of a cookie to a specific path on the server and can therefore be used to prevent unauthorized access to it from other applications on the same host.”

– **True or false?**

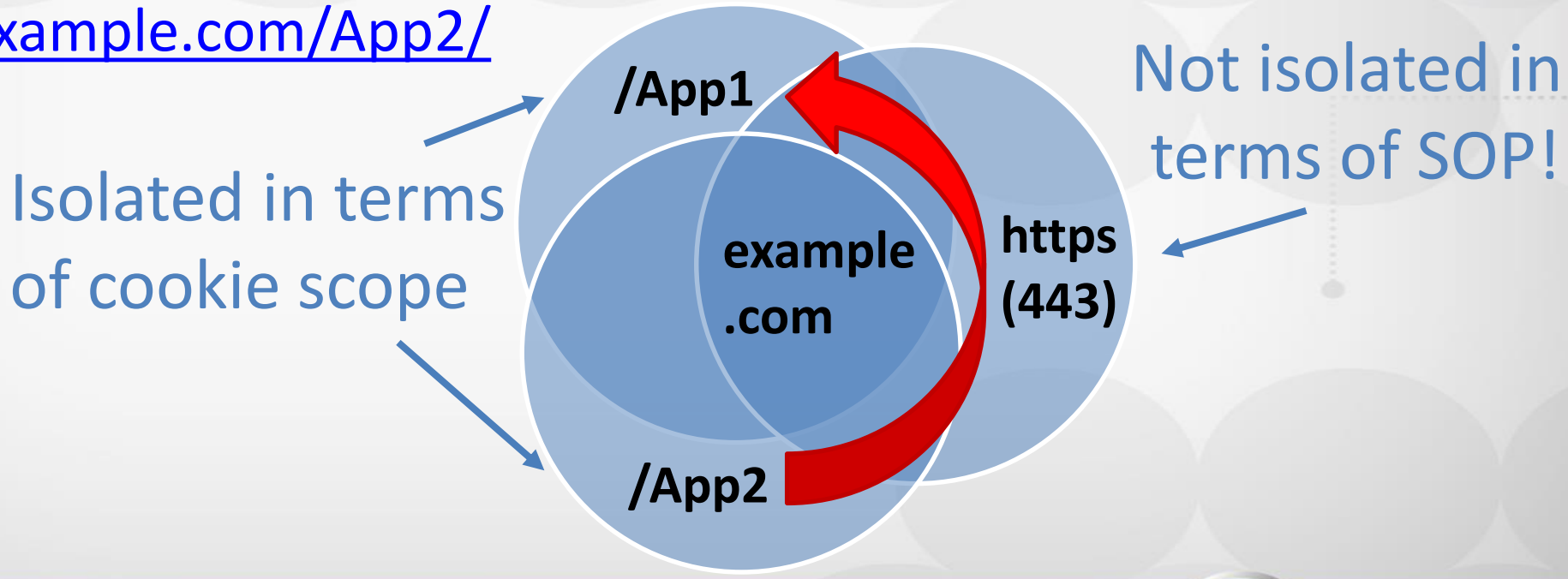
The ‘Path’ Attribute

- Cookie Scope vs. Same-origin Policy



The ‘Path’ Attribute

- Two different applications on shared host:
  - https://example.com/App1/
  - https://example.com/App2/



Only Send Cookie to Intended Host(s)

THE ‘DOMAIN’ ATTRIBUTE

The ‘Domain’ Attribute

“The ‘Domain’ attribute should be set to the origin host to limit the scope to that particular server. For example if the application resides on server app.mysite.com, then it should be set to domain=app.mysite.com”

– **True or false?**

The ‘Domain’ Attribute

- With domain set, cookies will be sent to that domain **and all its subdomains**
- The risk with subdomains is lower than when scoped to parent domain, but still relevant
- Remove domain attribute to limit cookie to origin host only
  - Important note: IE will always send to subdomains regardless

Limiting Exposure of Cookies

COOKIE LIFETIME

Cookie Lifetime

"A *session cookie*, also known as an *in-memory cookie* or *transient cookie*, exists only in temporary memory while the user navigates the website." (Wikipedia)

– **True or false?**

Cookie Lifetime

- It’s up to the browser to decide when the session ends
- ‘Non-persistent’ session cookies may actually be persisted to survive browser restart

> When user privacy is a concern, It is important that any web app implementation will invalidate cookie data after a certain timeout and won't rely on the browser clearing session cookies
>
> *One of the most beloved features of Firefox prevents session cookies from ever expiring.*
>
> *The same issue is also occuring with google chrome (and probably with other browsers offering similar features)*

https://developer.mozilla.org/en-US/docs/Web/API/document/cookie

RFC6265bis: Making Improvements to the Cookie Recipe

MODERN COOKIE PROTECTIONS

Strict Secure Cookies

- Makes ‘secure’ cookies a little more secure by adding integrity protection
- Prevents plain-text HTTP responses from setting or overwriting ‘secure’ cookies
- Attackers still have a window of opportunity to “pre-empt” secure cookies with their own

Cookie Prefixes

- Problem:
  - Server only sees cookie name and value in HTTP request, no information about its attributes
  - Impossible for server to know if a cookie it receives was set securely
- Solution:
  - ‘Smuggle’ information to server in cookie name
  - "__Secure-" prefix
  - "__Host-" prefix

The ‘SameSite’ Attribute

- Problem:
  - Cookies are sent with all requests to a server, regardless of request origin
  - Attackers can abuse this by initiating authenticated cross-origin requests, e.g., CSRF, XSSI, etc.
- Solution:
  - New cookie attribute SameSite=[Strict|Lax]
  - Prevents cookies from being attached to cross-origin requests

SUMMARY

Summary

- Key Takeaways:
  - Cookies are still largely based on a draft from 1994
  - The security model has many weaknesses
  - Don’t build your application on false assumptions about cookie security
  - Application and framework developers should take advantage of new improvements to cookie security
  - Beware that not all browsers are using the same cookie recipe (yet)

The ‘Ultimate’ Cookie

- Is there an ‘ultimate’ cookie configuration?
- This is probably the most secure configuration we have for now:

```
Set-Cookie: __Host-SessionID=3h93…;
Path=/;Secure;HttpOnly;SameSite=Strict
```

The End

Questions?

@securitybits