An updated security model of the web

Dr. Philippe De Ryck

https://Pragmatic Web Security.com

I am *Dr. Philippe De Ryck*

Pragmatic Web Security
Security for developers

Founder of Pragmatic Web Security

Google Developers Experts

Google Developer Expert

SecAppDev

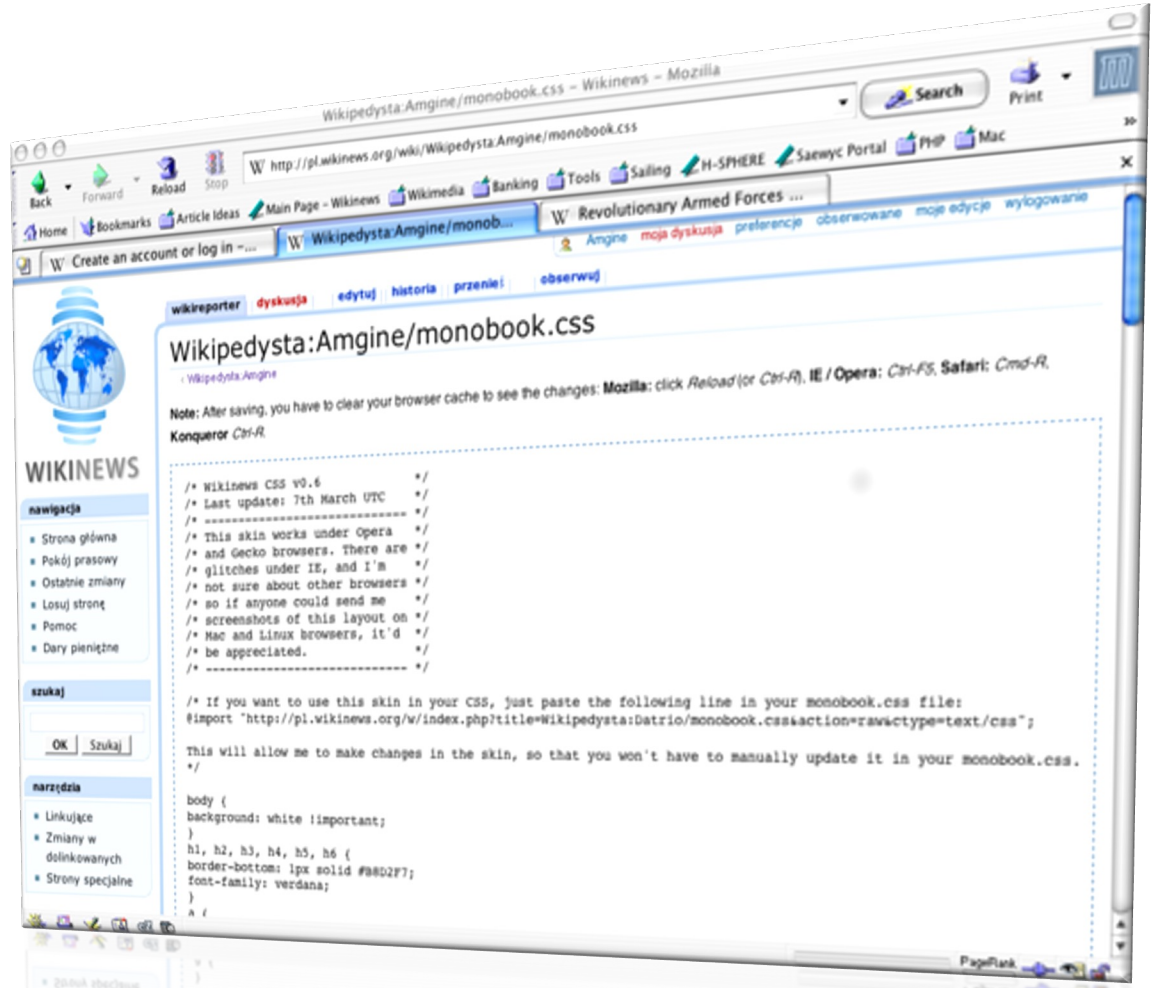
SecAppDev organizer

I help developers with security

- **Hands-on in-depth security training**
- **Advanced online security courses**
- **Security advisory services**

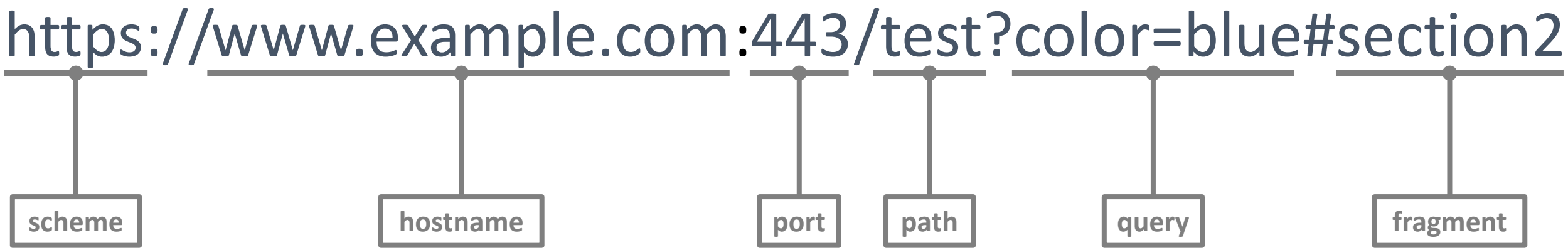
https://pdr.online


What's New
Check Email
YAHOO!
Personalize
Help
Yahoo! Auctions
coins, cards, stamps
Toshiba Laptop Giveaway!
ONSALEatCOST
Find a loan
Win $10,000
Search
advanced search
Yahoo! Games - Play online chess, bridge, spades, hearts and more...
Shopping - Yellow Pages - People Search - Maps - Travel Agent - Classifieds - Personals - Games - Chat
Email - Calendar - Pager - My Yahoo! - Today's News - Sports - Weather - TV - Stock Quotes - more...
Arts & Humanities
Literature, Photography...
Business & Economy
Companies, Finance, Jobs...
Computers & Internet
Internet, WWW, Software, Games...
Education
College and University, K-12...
Entertainment
Cool Links, Movies, Humor, Music...
Government
Military, Politics, Law, Taxes...
Health
Medicine, Diseases, Drugs, Fitness...
News & Media
Full Coverage, Newspapers, TV...
Recreation & Sports
Sports, Travel, Autos, Outdoors...
Reference
Libraries, Dictionaries, Quotations...
Regional
Countries, Regions, US States...
Science
Biology, Astronomy, Engineering...
Social Science
Archaeology, Economics, Languages...
Society & Culture
People, Environment, Religion...
In the News
Up to 25 dead in Colorado H.S. shooting
NATO - Serbia war
Year 2000 problem
more...
Marketplace
Kosovo Charity Auctions
Custom mortgage quotes at the Loan Center
more...
Inside Yahoo!
Y! Pager - instant messaging
Y! Clubs - something for everyone
Y! Calendar - your personal web calendar
more...
World Yahoo!s Europe : Denmark - France - Germany - Italy - Norway - Spain - Sweden - UK & Ireland
Pacific Rim : Australia & NZ - HK - Japan - Korea - Singapore - Taiwan -- Asia - Chinese
Americas : Canada - Spanish
Yahoo! Get Local LA - NYC - SF Bay - Chicago - more...
Enter Zip Code

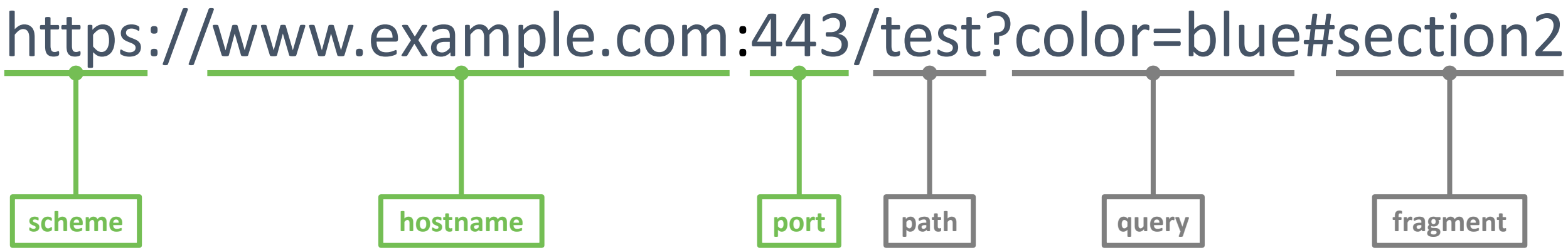

Wikipedysta:Amgine/monobook.css – Wikinews – Mozilla
Search
Print
Back
Forward
Reload
Stop
http://pl.wikinews.org/wiki/Wikipedysta:Amgine/monobook.css
Bookmarks
Article Ideas
Main Page – Wikinews
Wikimedia
Banking
Tools
Sailing
H-SPHERE
Saewyc Portal
PHP
Mac
Home
Create an account or log in – ...
Wikipedysta:Amgine/monob...
Revolutionary Armed Forces ...
Amgine
moja dyskusja
preferencje
obserwowane
moje edycje
wylogowanie
wikireporter
dyskusja
edytuj
historia
przenieś
obserwuj
Wikipedysta:Amgine/monobook.css
Wikipedysta:Amgine
Note: After saving, you have to clear your browser cache to see the changes: Mozilla: click Reload (or Ctrl-R), IE / Opera: Ctrl-F5, Safari: Cmd-R, Konqueror Ctrl-R.
WIKINEWS
nawigacja
Strona główna
Pokój prasowy
Ostatnie zmiany
Losuj stronę
Pomoc
Dary pieniężne
szukaj
OK
Szukaj
narzędzia
Linkujące
Zmiany w dolinkowanych
Strony specjalne
/* Wikinews CSS v0.6 */
/* Last update: 7th March UTC */
/* This skin works under Opera */
/* and Gecko browsers. There are */
/* glitches under IE, and I'm */
/* not sure about other browsers */
/* so if anyone could send me */
/* screenshots of this layout on */
/* Mac and Linux browsers, it'd */
/* be appreciated. */
/* If you want to use this skin in your CSS, just paste the following line in your monobook.css file:
@import "http://pl.wikinews.org/w/index.php?title=Wikipedysta:Datrio/monobook.css&action=raw&ctype=text/css";
This will allow me to make changes in the skin, so that you won't have to manually update it in your monobook.css.
*/
body {
background: white !important;
}
h1, h2, h3, h4, h5, h6 {
border-bottom: 1px solid #88D2F7;
font-family: verdana;
}
PageRank

Origins

THE DEFINITION OF AN ORIGIN

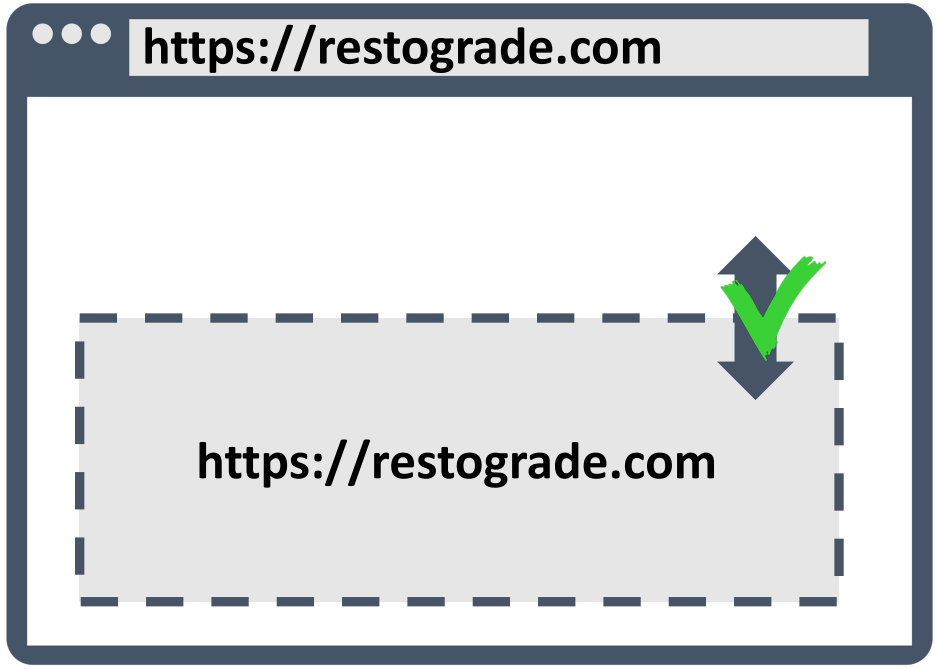


THE DEFINITION OF AN ORIGIN



THE *SAME-ORIGIN POLICY (SOP)*

Content retrieved from one origin can freely interact with other content from that origin, but interactions with content from other origins are restricted

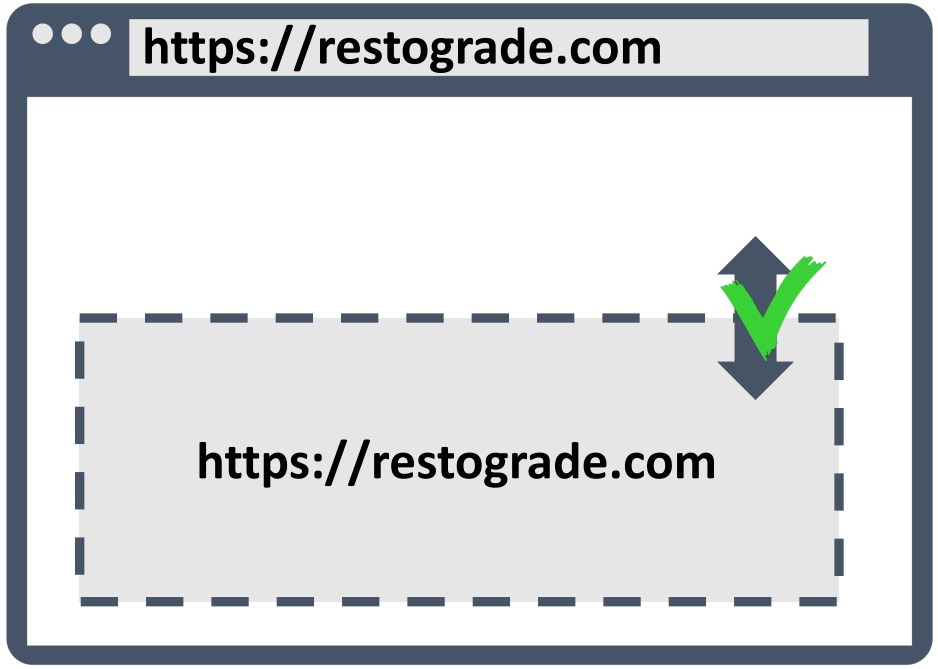


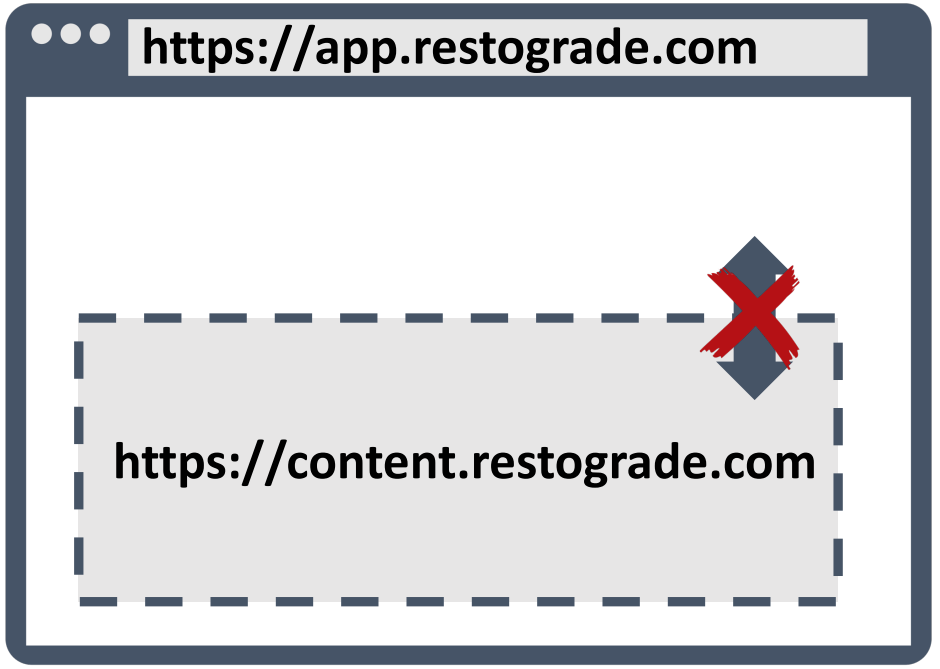
Loading an iframe in an HTML page

```
1 <iframe src="https://restograde.com">
2 </iframe>
```

THE *SAME-ORIGIN POLICY (SOP)*

Content retrieved from one origin can freely interact with other content from that origin, but interactions with content from other origins are restricted





THE ORIGIN AS A SECURITY PRINCIPAL

- Origins are used as a principal for making security decisions
  - The Same-Origin Policy governs interaction between contexts
  - The SOP affects the DOM and all its contents
- Other origin-protected resources in a modern browser
  - Permissions for sensitive features are also granted per origin
  - Client-side storage areas (Web Storage, IndexedDB, ...)
  - Ability to send JavaScript-based XHR requests without CORS restrictions
    - Includes the capability to load resources and inspect their contents (e.g. JS source code)
- One of the most important aspects of web security is controlling your origin
  - Once an attacker runs code within your origin, it will be hard to provide any security

LEVERAGING ORIGIN ISOLATION FOR SECURITY

example.com/calendar

example.com/forum

example.com/admin

Browsers cannot isolate based on paths, so each of these applications runs in same "trust zone". One piece of malicious JS code in any of these apps can influence all the other apps.

calendar.example.com

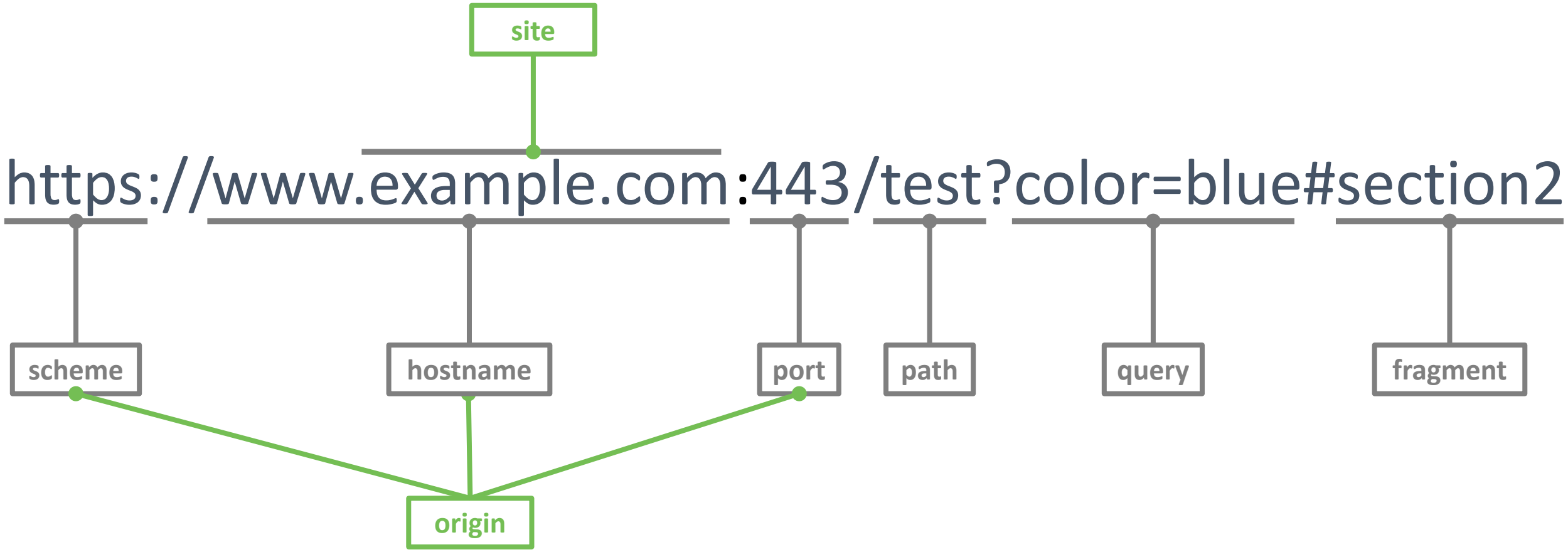
forum.example.com

admin.example.com

The suite stills run within the same site, but each of the applications runs in an isolated origin. A vulnerability in the forum will not automatically affect the admin app.

Sites

ORIGINS AND SITES




Cross-Site
https://bruce.github.io
https://batman.github.io
github.io is registered as an eTLD, so the sites are respectively bruce.github.io and batman.github.io

The site is derived from the hostname and consists of the eTLD + 1

The eTLD is determined based on the public suffix list, which is hardcoded in browsers

https://publicsuffix.org/

ORIGINS AND SITES

- Same-site or cross-site is determined based on the ***eTLD + 1***
  - For traditional eTLDs, this corresponds to the registered domain
    - I.e., what you buy from a registrar (E.g., *restograde.com*, *restograde.co.uk*)
  - The proper way to determine the eTLD is by checking the public suffix list
    - Domains can be registered as a TLD (e.g., *github.io*)

- Everything running in a site is considered to (loosely) belong together
  - The browser still enforces the Same-Origin Policy on each individual browsing context
  - Additional security measures sometimes rely on the cross-site property of a context
    - E.g., the cookie *SameSite* flag, *Cross-Origin Resource Policy (CORP)* with a same-site config

- Subdomains are considered cross-origin but not cross-site
  - Avoid giving control of subdomains to untrusted or external parties

Which of these URLs are cross-origin compared to *https://app.restograde.com/calendar/*

A `http://app.restograde.com/calendar/`

B `https://app.restograde.com/reviews/`

C `https://app.restograde.com:8443/calendar/`

D `https://www.restograde.com/calendar/`

Which of these URLs are cross-site compared to *https://app.restograde.com/calendar/*

A `https://www.restograde.com/calendar/`

B `https://reviews.restograde.com/`

C `https://restogradecalendar.com/`

D None of the above

Chromium-based browsers define a site as the tuple *<scheme, eTLD + 1>* to distinguish between HTTP and HTTPS sites

headers HTTP header: Set-Cookie: SameSite: URL scheme-aware ("schemeful")

Usage % of all users ?

Global 76.34%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ! *
4-90	12-90		2-78	10-76	
91-147	91-147	3.1-26.3	1 79-150	77-127	6-10
148	148	26.4	1 151	131	11
149-151		26.5-TP	1 152-154		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile
		4-15.0	
	3.2-26.3	16.0-28	12-12.1
148	26.4	29	80
	26.5		

Since everything you build runs over HTTPS anyway, this has very little impact

LEVERAGING SITE ISOLATION FOR SECURITY

calendar.example.com

forum.example.com

admin.example.com

These resources are isolated in their own origin, but all belong to the same site (example.com). As a result, the browser can allow laxer policies for same-site configurations.

examplecalendar.com

exampleforum.com

exampleadmin.com

Each application now has its own site, ensuring full origin-based and site-based isolation in the browser. This works well for unrelated applications, which can easily be deployed in different sites.

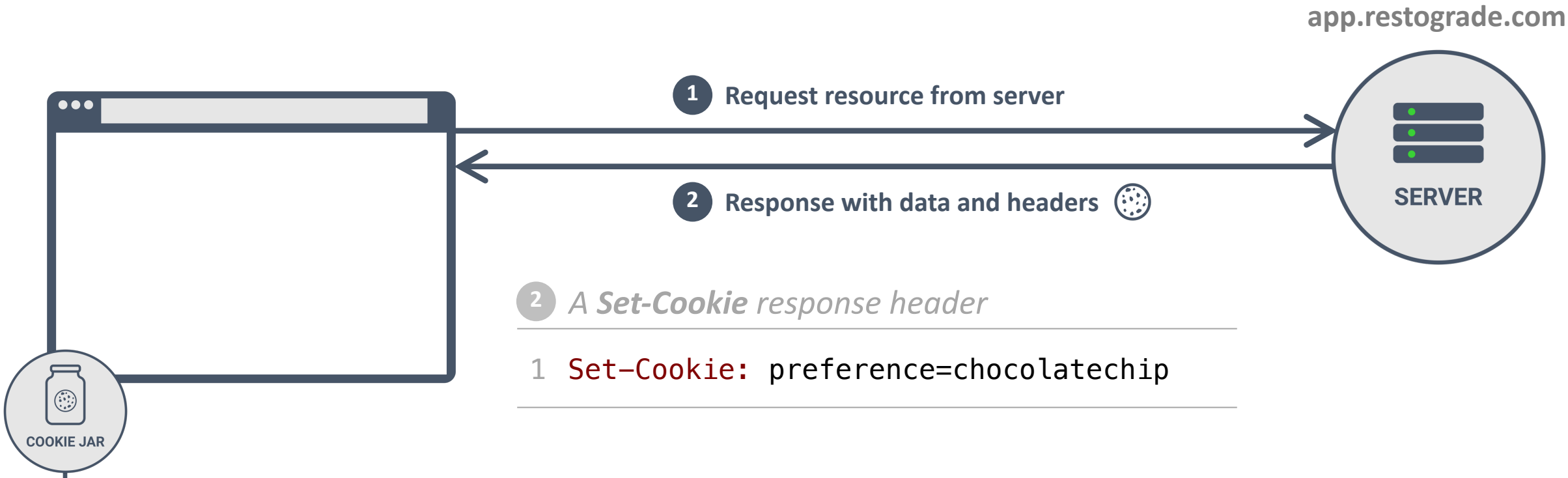
In the previous example, we can also register *example.com* as an eTLD, making all subdomains cross-site from each other

Using customized eTLDs is possible and legitimate, but takes a solid understanding of the potential impact. This is an advanced feature.

Cookies

Cookies used to be the de facto mechanism to keep track of authentication state

Even in modern applications, cookies are still a valid option. Additionally, modern security patterns like the OAuth 2.0 Backend-For-Frontend rely on cookies.



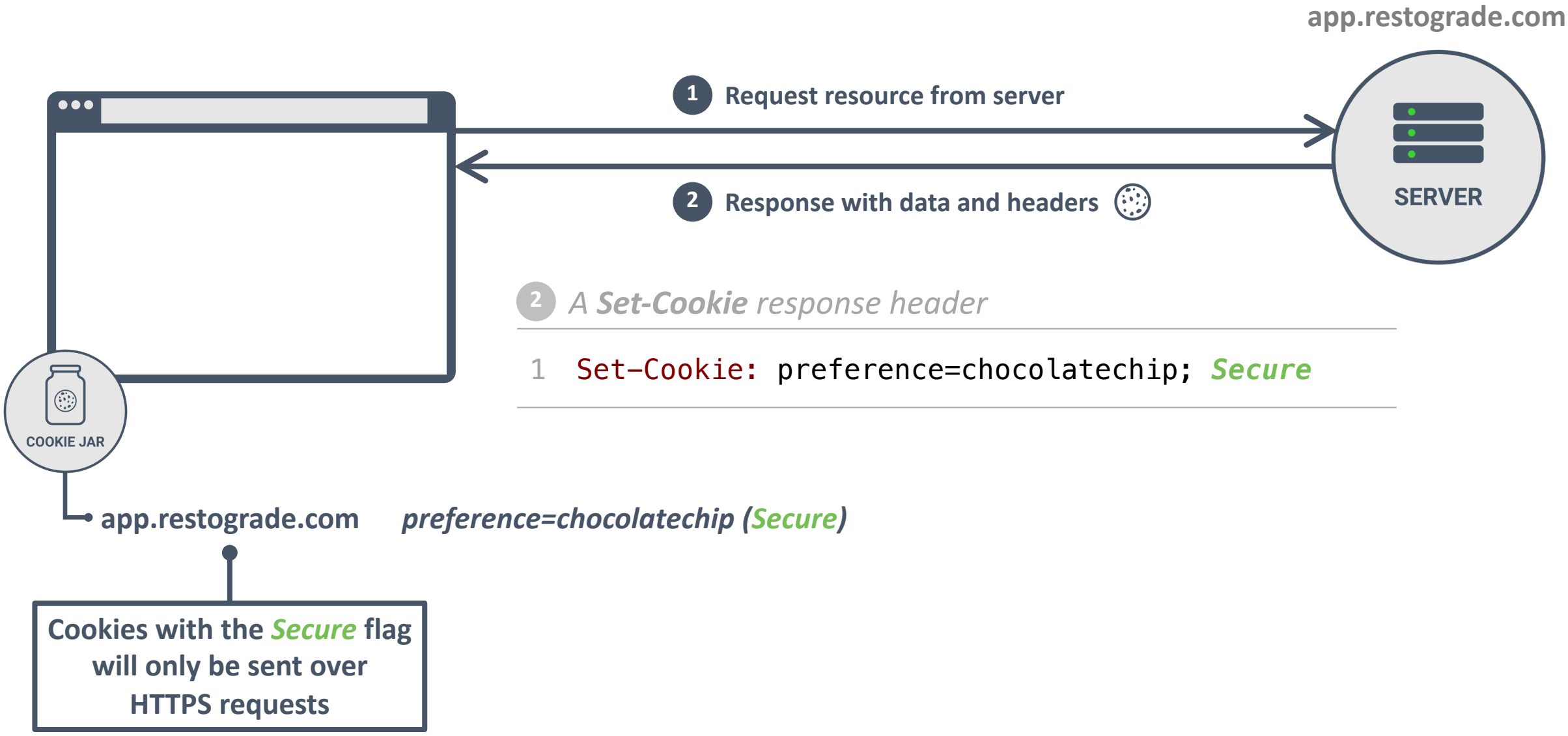
2 *A **Set-Cookie** response header*

```
1 Set-Cookie: preference=chocolatechip
```

app.restograde.com *preference=chocolatechip*

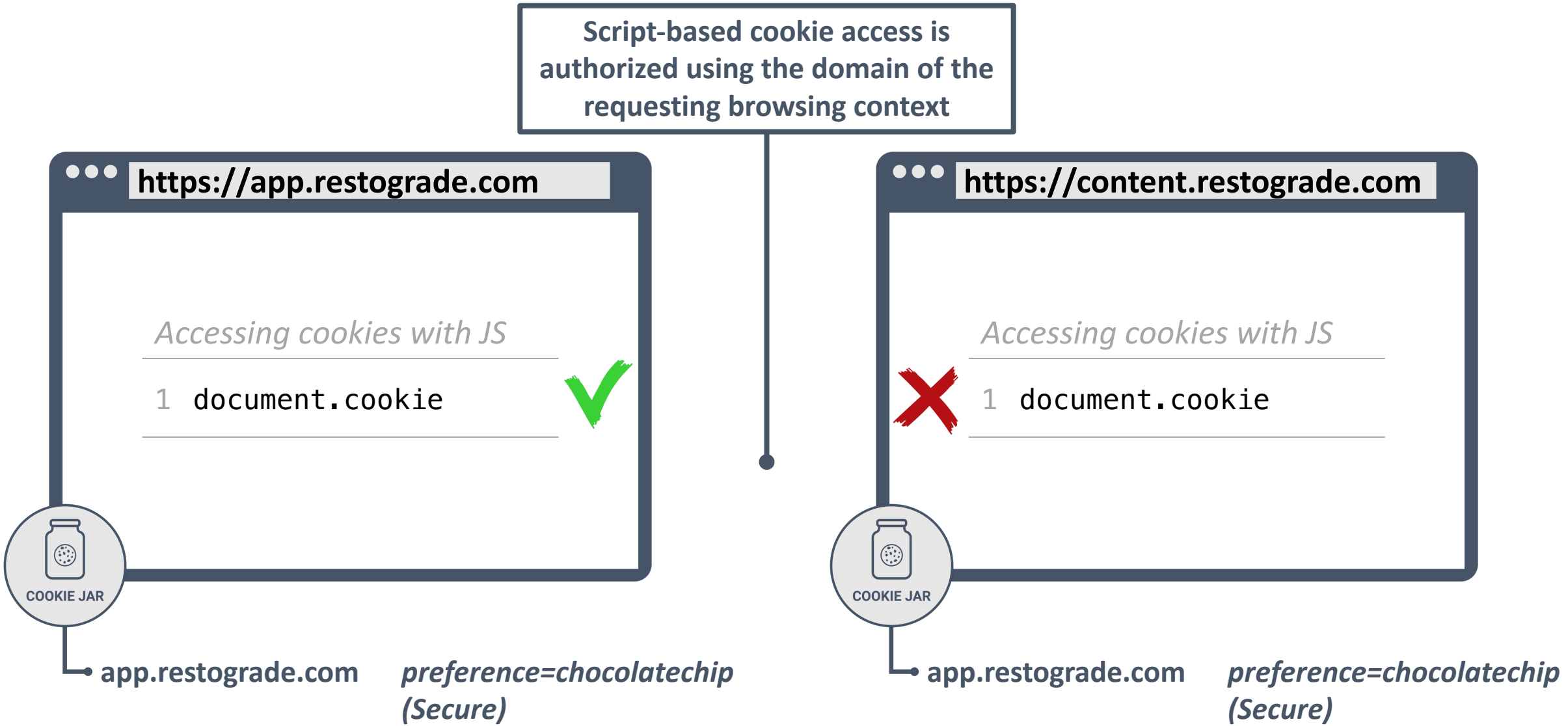
Cookies are only associated with a *domain*, not with a scheme or a path

The browser automatically attaches the cookie on outgoing requests to app.restograde.com



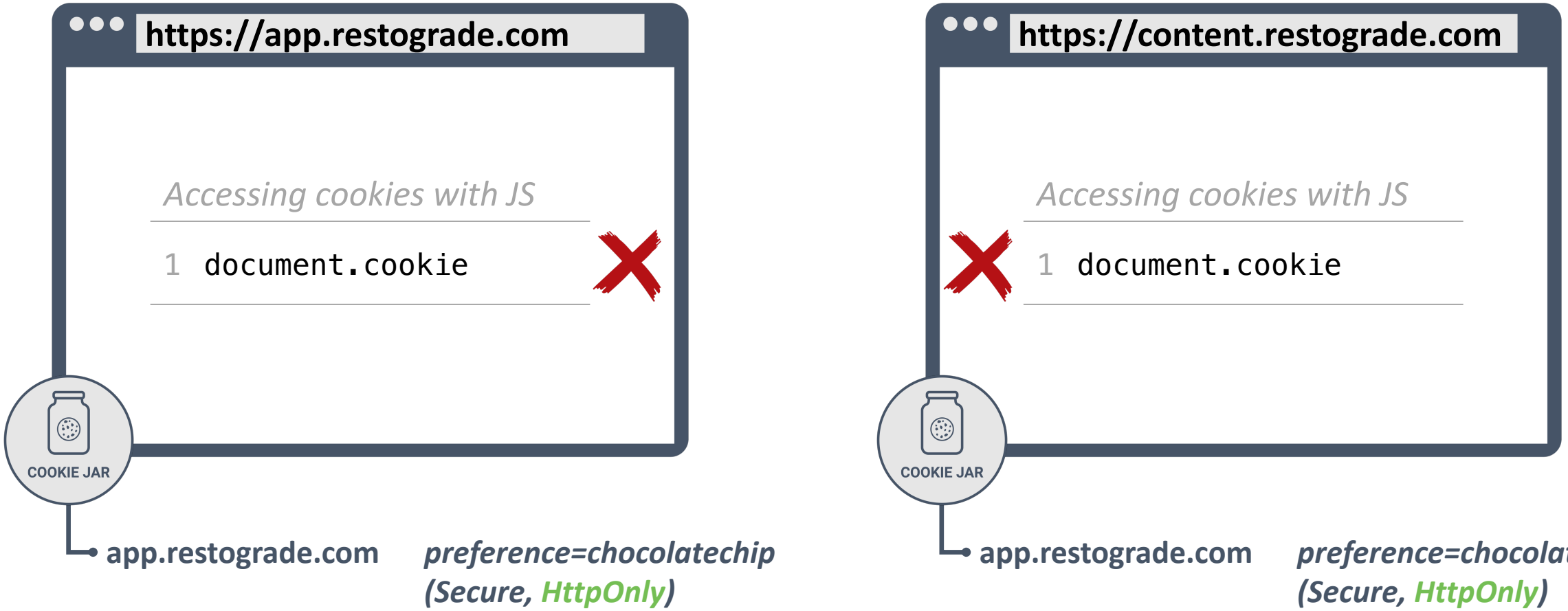
2 *A **Set-Cookie** response header*

```
1  Set-Cookie: preference=chocolatechip; Secure
```



*A **Set-Cookie** response header*

```
1 Set-Cookie: preference=chocolatechip; Secure
```



*A **Set-Cookie** response header*

```
1  Set-Cookie: preference=chocolatechip; Secure; HttpOnly
```

The *HttpOnly* flag tells the browser to not expose the cookie to JavaScript

What is a cookie tossing attack?

2 *A **malicious** Set-Cookie response header*

```
1  Set-Cookie: JSESSIONID=AttackerSession; Domain=restograde.com
```

Cookie tossing is possible if the attacker can manipulate insecure HTTP responses (less common these days)

An alternative cookie tossing attack vector is a subdomain takeover, where the attacker can set domain-wide cookies from a compromised subdomain

Rampant CNAME misconfiguration leaves thousands of organizations open to subdomain takeover attacks – research

Adam Bannister 25 November 2020 at 14:46 UTC
Updated: 18 May 2021 at 13:46 UTC

DNS Browsers Vulnerabilities

Security researchers discover more than 400,000 at-risk subdomains during an automated internet trawl

https://portswigger.net/daily-swig/rampant-cname-misconfiguration-leaves-thousands-of-organizations-open-to-subdomain-takeover-attacks-nbsp-research

Cookie prefixes help protect against cookie tossing attacks

The __Secure- cookie prefix

- The name of the cookie can be prefixed with ***__Secure-***
  - The cookie can only be set over a secure connection
  - The cookie can only be set with the ***Secure*** flag enabled

- Since the ***__Secure-*** prefix is part of the name, it is sent to the server
  - The server now knows that the cookie has been set over HTTPS
  - Whoever set the cookie was able to set up a valid HTTPS connection

- Attackers able to set such prefixed cookies can do a lot worse


```
Set-Cookie: __Secure-session=...; Secure; HttpOnly
```


```
Cookie: __Secure-session=...
```

THE __HOST- COOKIE PREFIX

- The name of the cookie can also be prefixed with __*Host-*
  - Everything from the __*Secure-* prefix applies
  - The cookie can only be set for the root path (/)
  - The cookie will only be sent to that host, never for sibling or child domains

- Since the __*Host-* prefix is part of the name, it is sent to the server
  - Whoever set the cookie was able to set up a valid HTTPS connection for the domain

- Attackers able to set a __*Host-* have full control of the application


```
Set-Cookie: __Host-session=...; Secure; HttpOnly
```


```
Cookie: __Host-session=...
```

THE __HTTP- COOKIE PREFIX

- The name of the cookie can also be prefixed with __*Http-*
  - Everything from the __*Secure-* prefix applies
  - The cookie must have the ***HttpOnly*** flag set, otherwise it will not be accepted
  - As a result, this cookie cannot be set or read by JavaScript

- Attackers able to set a __*Http-* cookie must do so in a network response
  - The cookie cannot be set from JavaScript in any way
  - To manipulate network responses, the attacker needs to break HTTPS
  - The attacker can still run a subdomain-based attack with domain-wide cookies


```
Set-Cookie: __Http-session=...; Secure; HttpOnly
```


```
Cookie: __Http-session=...
```

THE __HOST-HTTP- COOKIE PREFIX

- The name of the cookie can also be prefixed with __*Host-Http-*
  - Everything from the __*Host-* prefix applies
  - The cookie must have the ***HttpOnly*** flag set, otherwise it will not be accepted
  - As a result, this cookie cannot be set or read by JavaScript

- The __*Host-Http-* prefix makes the cookie origin-bound without JS access
  - This cookie cannot be set from any other host or domain
  - This cookie cannot be read or set from JS


```
Set-Cookie: __Host-Http-session=...; Secure; HttpOnly
```


```
Cookie: __Host-Http-session=...
```

headers HTTP header: Set-Cookie: Cookie prefixes: `__Host-` and `__Secure-`

Usage % of all users ?

Global 95.36%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ! *
4-48	12-18	3.1-12.1	2-49	10-35	
49-147	79-147	13-26.3	50-150	36-127	6-10
148	148	26.4	151	131	11
149-151		26.5-TP	152-154		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile
	3.2-12.5	4	

headers HTTP header: Set-Cookie: Cookie prefixes: `__Http-` and `__Host-Http-`

Usage % of all users ?

Global 68.61%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ! *
			2-141		
4-139	12-139		1 142	10-123	
140-147	140-147	3.1-26.3	143-150	124-127	6-10
148	148	26.4	151	131	11
149-151		26.5-TP	152-154		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile
	3.2-26.3	4-28	12-12.1
148	26.4	29	80
	26.5		

What happens on Safari with __*Host-Http*?

The __*Host-Http-* prefix is backwards compatible with the __*Host-* prefix

COOKIE BEST PRACTICES

- Cookies are associated with a domain instead of an origin
  - The use of the *Domain* attribute allows cookies to be used on multiple subdomains
    - Hard to control, so recommended to avoid this property if possible
  - The use of the *Path* attribute allows cookie separation per path
    - Mainly useful for cleanliness, ***not a security measure***

- Cookies can be configured with additional security properties
  - ***Secure*** restricts cookies to HTTPS only
  - ***HttpOnly*** prevents JS-based cookie access
  - The ***__Secure-***, ***__Host-*** , ***__Http-***, or ***__Host-Http-*** prefixes enable more secure handling

The current recommended best practice for sensitive cookies

```
1  Set-Cookie: __Host-Http-session=1a2b3c4d; Secure; HttpOnly
```

Preventing DOM-based XSS


BROWSER
3 API call to request data
1 Load webpage from server
2 Serve webpage with HTML and JS
4 Data response
BACKEND
5 Render the data containing *attacker-provided data*
6 Execute *malicious payload* in the application's execution context

5 Render the data containing *attacker-provided data*

6 **Execute *malicious payload* in the application's execution context**

3 Render *untrusted* data coming from the browser (e.g., URL, postMessage, …)

4 **Execute *malicious payload* in the application's execution context**

MITIGATING DOM-BASED XSS

- The root cause of XSS is a lack of context when the browser processes the data
  - Caused by the mismatch between server-side composition and client-side rendering
  - But now we're injecting data into the DOM within the browser ...

- The best defense against DOM-based XSS is to provide proper context info
  - Use safe DOM APIs to inject data into the page

Giving the browser a snippet of data without context information

```
1  document.getElementById("msg").innerHTML = e.data
```

innerHTML relies on the browser's code parser to handle the data

Giving the browser a snippet of data with context information

```
1  document.getElementById("msg").textContent = e.data
```

textContent tells the browser this is text data, not code

DOM-based XSS is a misnomer.
Client-side XSS is a better description.

Client-side XSS in action

MITIGATING DOM-BASED XSS

- The root cause of XSS is a lack of context when the browser processes the data
  - Caused by the mismatch between server-side composition and client-side rendering
  - But now we're injecting data into the DOM within the browser ...

- The best defense against DOM-based XSS is to provide proper context info
  - Use safe DOM APIs to inject data into the page
    - E.g., ***textContent*** instead of ***innerHTML***
    - E.g., ***createElement(…)*** and ***setAttribute(…)*** instead of ***innerHTML***
  - Avoid the use of ***innerHTML***, ***outerHTML***, ***write***, ***writeln***

- Client-side output that requires benign HTML can rely on sanitization
  - Use *DOMPurify* to sanitize the output in the browser
  - Output the data into the page with *innerHTML*

Mitigating client-side XSS with sanitization

Setting HTML of an element, instead of using ***innerHTML***

Transforming text into an HTML document, instead of using ***DOMparser***

myDOMelement.setHTML(unsafeString)

Document. parseHTML(unsafeString)

myDOMelement. setHTML(unsafeString, sanitizer)

Document. parseHTML(unsafeString , sanitizer)

You can configure a custom sanitizer, which allows/disallows specific elements and attributes.
Only use this when you know what you are doing, since it can re-introduce XSS vulnerabilities

Creating an "allow" configuration

This example shows how you might create an "allow" sanitizer configuration that allows specific elements and attributes, replaces `<b>` elements with their children, allows comments to be included in the output, and requires that `data-*` attributes are explicitly listed in the `attributes` array to be included. The configuration object is passed to the `Sanitizer()` constructor.

JS Copy

```
const sanitizer = new Sanitizer({
  elements: ["div", "p", "script"],
  attributes: ["id"],
  replaceWithChildrenElements: ["b"],
  comments: true,
  dataAttributes: false,
});
```

https://developer.mozilla.org/en-US/docs/Web/API/SanitizerConfig#examples

Setting HTML of an element, instead of using *innerHTML*

Transforming text into an HTML document, instead of using *DOMparser*

myDOMelement. setHTML(unsafeString)

Document. parseHTML(unsafeString)

myDOMelement. setHTML(unsafeString, sanitizer)

Document.parseHTML(unsafeString , sanitizer)

You can configure a custom sanitizer, which allows/disallows specific elements and attributes. Only use this when you know what you are doing, since it can re-introduce XSS vulnerabilities

myDOMelement.setHTMLUnsafe(unsafeString)

Document.parseHTMLUnsafe(unsafeString)

Applies no sanitization on the input, making it dangerous on untrusted input. Rarely useful.

myDOMelement. setHTMLUnsafe(unsafeString , sanitizer)

Document.parseHTMLUnsafe(unsafeString , sanitizer)

Applies the sanitizer on the input. This use of unsafe is for when you do partial sanitization, as best as possible, but know that the output is still not considered safe.

Sanitizer API

Usage % of all users ?

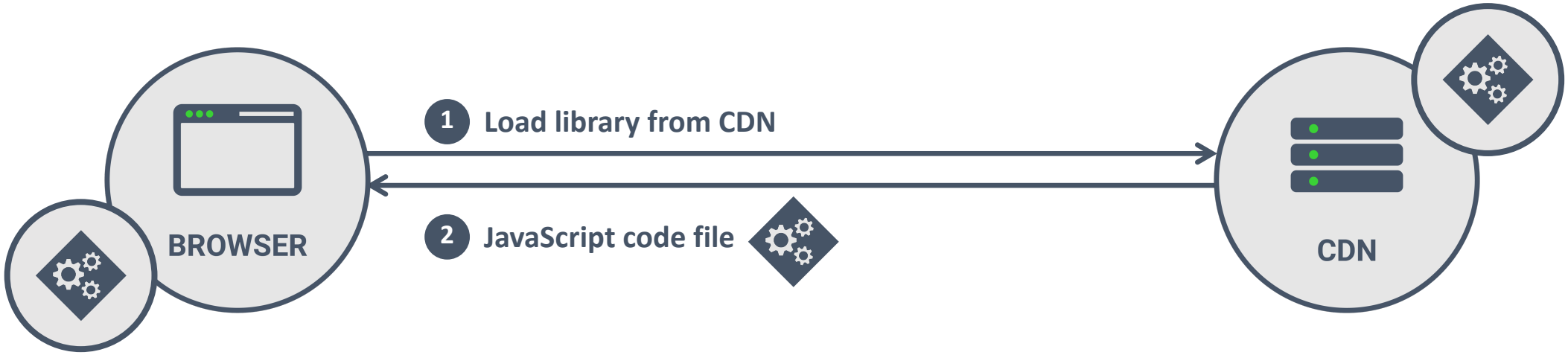
Global 65.89%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ⚠ *	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile
4-145	12-145		2-147						
1 146-147	2 146-147	3.1-26.3	148-150	10-127	6-10		3.2-26.3	4-28	12-12.1
1 148	2 148	26.4	151	3 131	11	4 148	26.4	29	80
1 149-151		26.5-TP	152-154				26.5		

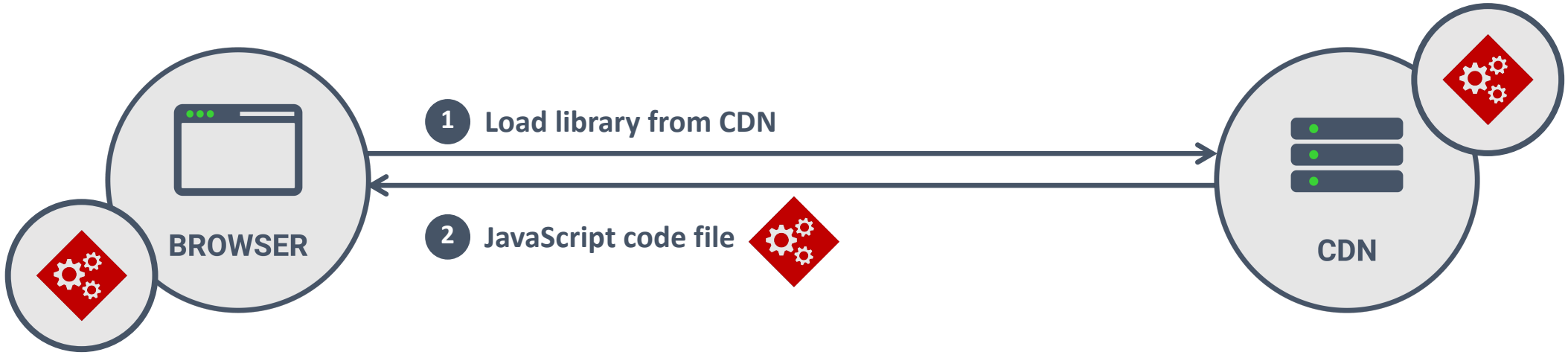
A polyfill can be used until Safari gets their shit together and implements this API

Integrity protection for remote script files



Loading a JavaScript file from a CDN

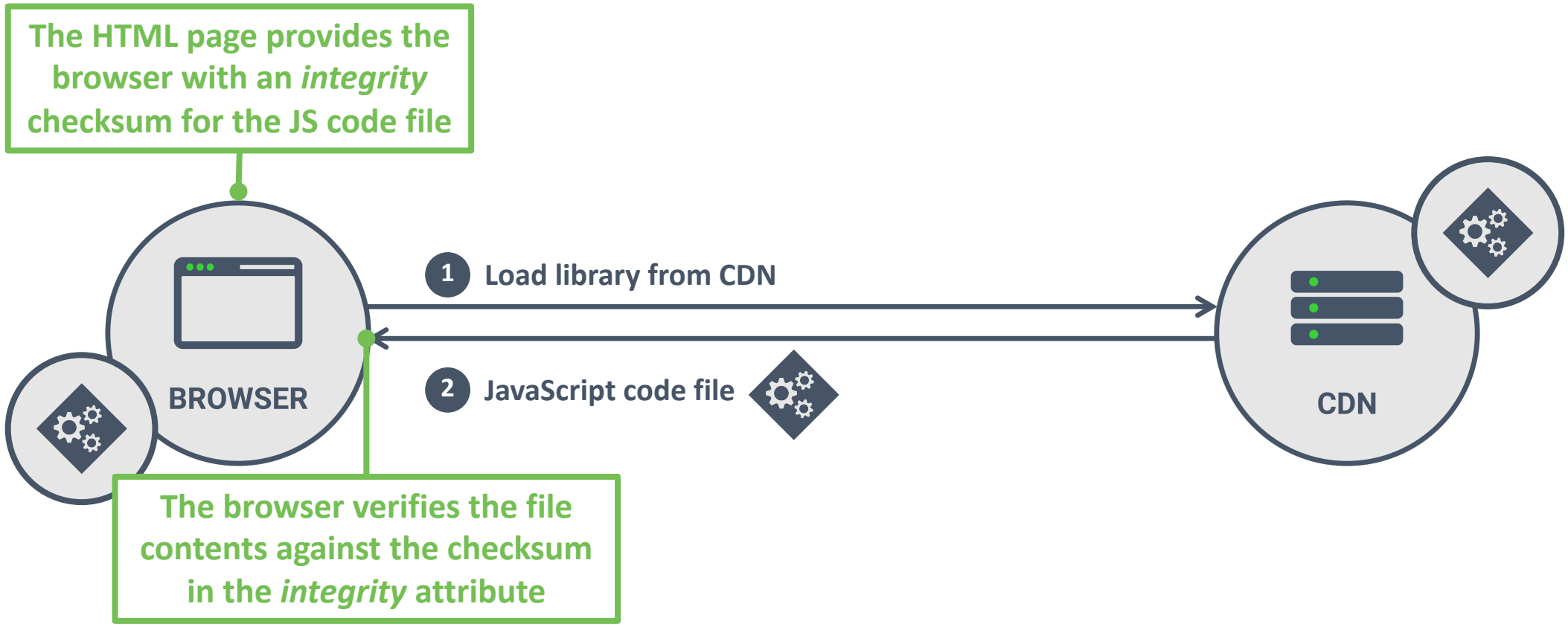
```
1  <script src="https://...cdn.../angular_1.7.js">
2  </script>
```



Loading a JavaScript file from a CDN

```
1  <script src="https://...cdn.../angular_1.7.js">
2  </script>
```

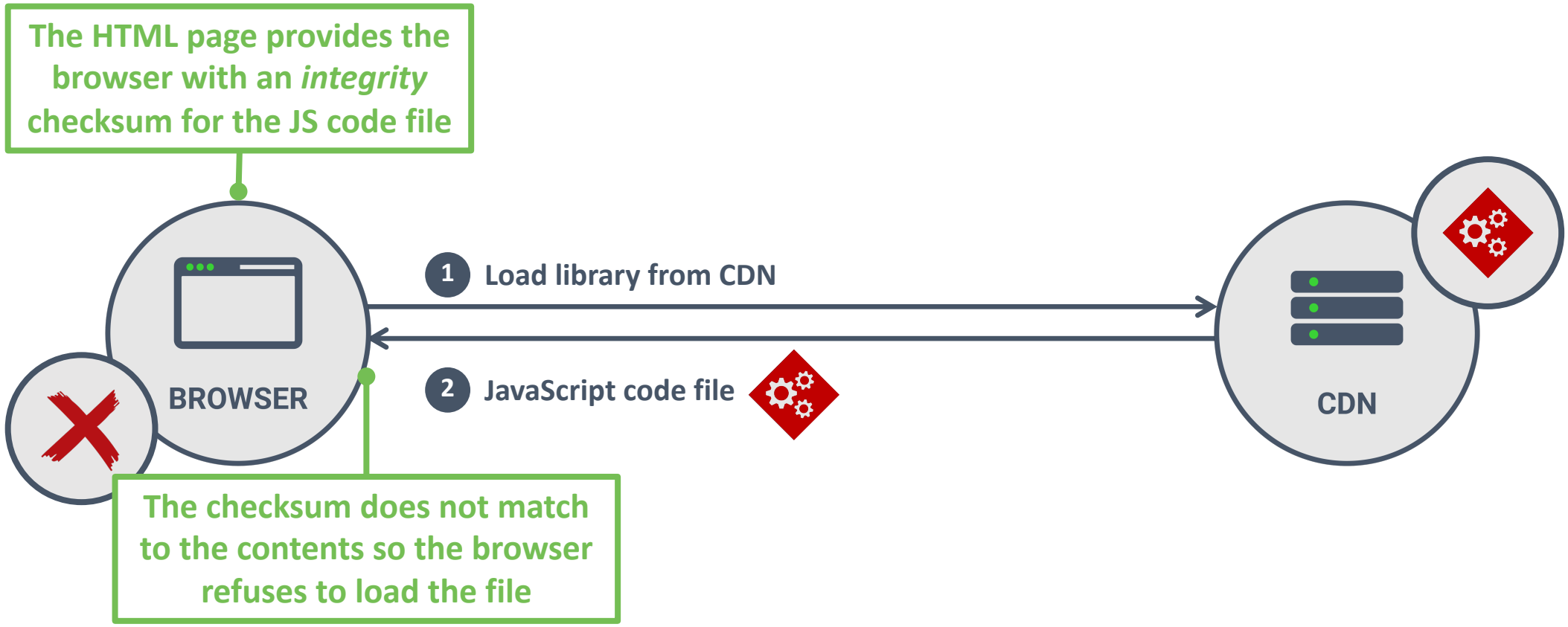
SUBRESOURCE INTEGRITY



Loading a JavaScript file from a CDN

```
1  <script src="https://...cdn.../angular_1.7.js"
2          integrity="sha384-ChfqqxuZUCnJ...JmZQ5stwEULTy">
3  </script>
```

SUBRESOURCE INTEGRITY



Loading a JavaScript file from a CDN

```
1  <script src="https://...cdn.../angular_1.7.js"
2          integrity="sha384-ChfqqxuZUCnJ...JmZQ5stwEULTy">
3  </script>
```

Subresource Integrity - REC

Usage % of all users ?

Global 96.05%

Subresource Integrity enables browsers to verify that file is delivered without unexpected manipulation.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *
							3.2-10.3			
4-44	12-16	3.1-10.1	2-42	10-31			1 11.2	4		
45-135	17-135	11-18.4	43-137	32-116	6-10		11.3-18.4	5-26		12-12.1
136	136	18.5	138	117	11	136	18.5	27	all	80
137-139		TP	139-141							

SRI makes perfect sense in a secure web, so you should make it mandatory (if you can!)

Integrity-Policy header

The HTTP `Integrity-Policy` response header allows website administrators to ensure that all resources the user agent loads (of a certain type) have Subresource Integrity guarantees.

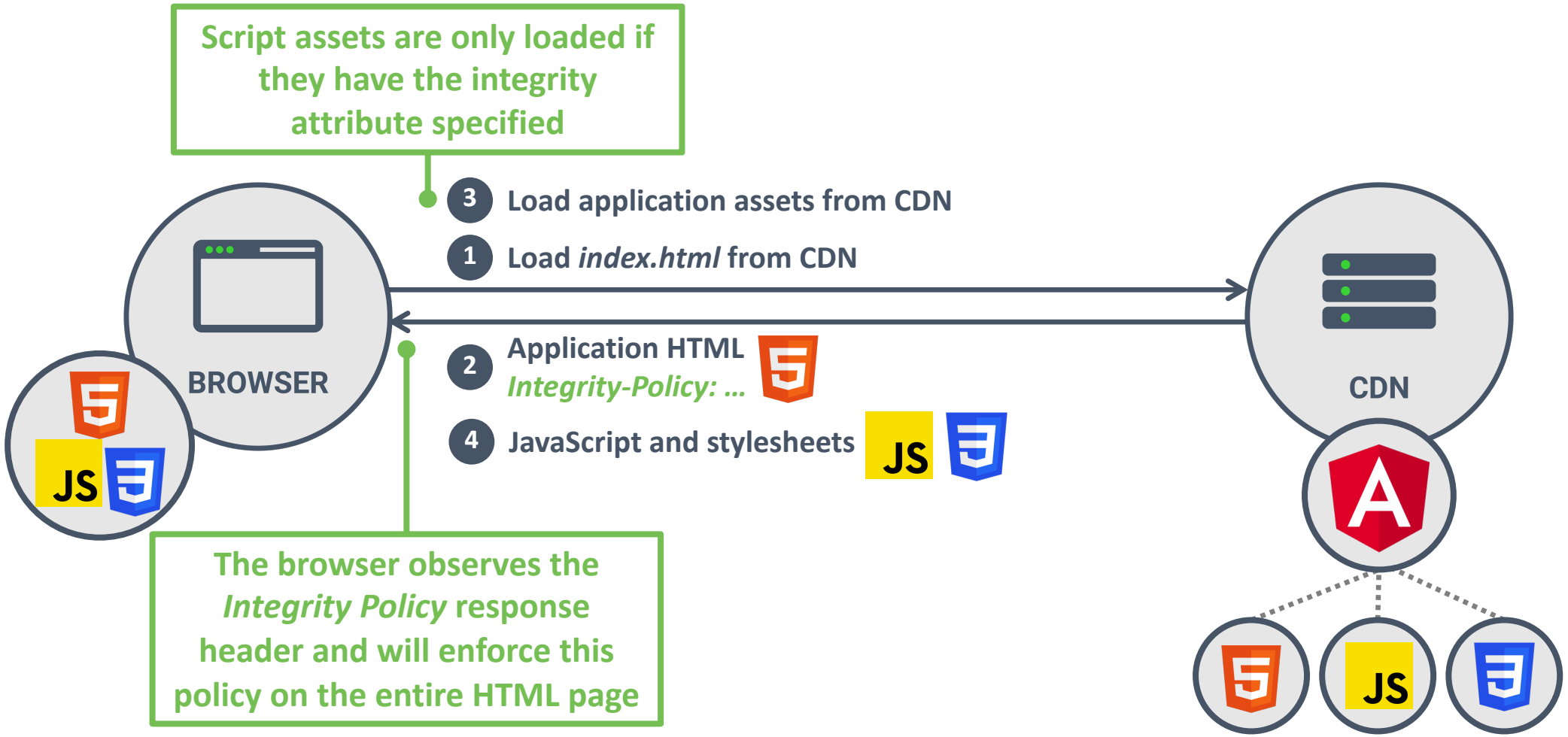
When set the user agent will block requests on specified request destinations that omit integrity metadata, and will also block requests in no-cors mode from ever being made.

Violation reports may also be sent to if the header includes a reporting endpoint name that matches an endpoint declared using the `Reporting-Endpoints` header. Reports are generated using the Reporting API, and may also be observed in the page for which the integrity policy is being enforced, using a `ReportingObserver`. The format of the report body is given by the `IntegrityViolationReportBody` dictionary (a JSON-serialized form of this body is sent in POSTs to reporting server endpoints).

This helps guard against content manipulation of fetched subresources.

Header type	Response header

In this article



A sample Integrity Policy *header configuration*

```
Integrity-Policy: blocked-destinations=(script)
```

headers HTTP header: Integrity-Policy

Usage % of all users ?

Global 78.1% + 1.87% = 79.97%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ⚠ *
4-137	12-137	3.1-18.7	2-144	10-121	
138-147	138-147	26.0-26.3	1 145-150	122-127	6-10
148	148	26.4	1 151	131	11
149-151		26.5-TP	1 152-154		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile	UC Browser for Android
	3.2-18.7			
	26.0-26.3	4-28	12-12.1	
148	26.4	29	80	15.5
	26.5			

A sample Integrity Policy *header configuration with reporting enabled*

```
Reporting-Endpoints: integrity-endpoint="https://restograde.com/reports/integrity"
Integrity-Policy:    blocked-destinations=(script),
                     endpoints=(integrity-endpoint)
```

The *Integrity Policy* can be configured to send violation reports to a reporting endpoint

Reports are sent out-of-band, so it may take a while for them to show up

The report contains all relevant information about the violation that occurred in the user's browser

A sample Integrity Policy *report*

```
{
  "type": "integrity-violation",
  "url": "https://restograde.com",
  "body": {
    "documentURL": "https://restograde.com",
    "blockedURL": "https://cdn.restograde.com/mylib.js",
    "destination": "script",
    "reportOnly": false
  }
}
```

A sample Integrity Policy *header configuration in* ***report only*** *mode*

```
Reporting-Endpoints: integrity-endpoint="https://restograde.com/reports/integrity"
Integrity-Policy-Report-Only: blocked-destinations=(script),
                              endpoints=(integrity-endpoint)
```

The *Integrity Policy* can be configured as report only to monitor violations but not block resources from being loaded

A sample Integrity Policy *report for report only policies*

```
{
  "type": "integrity-violation",
  "url": "https://restograde.com",
  "body": {
    "documentURL": "https://restograde.com",
    "blockedURL": "https://cdn.restograde.com/mylib.js",
    "destination": "script",
    "reportOnly": true
  }
}
```

Report Only information is great to try out a policy or to monitor potentially problematic behavior without breaking anything

headers HTTP header: Reporting-Endpoints

Usage % of all users ?

Global 94.15%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE ⚠ *
4-95	12-95	3.1-16.3	2-129	10-81	
96-147	96-147	16.4-26.3	130-150	82-127	6-10
148	148	26.4	151	131	11
149-151		26.5-TP	152-154		

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile	UC Browser for Android
	3.2-16.3	4-16.0		
	16.4-26.3	17.0-28	12-12.1	
148	26.4	29	80	15.5
	26.5			

Making SRI mandatory with Integrity Policy

- The ***Integrity Policy*** is specified as a response header on an HTML resource
  - The browser enforces the Integrity Policy on subresources of this HTML page
  - Each HTML response can have its own Integrity Policy definition

- Making SRI mandatory works great for applications relying on static script files
  - The use of SRI on static script files is a security best practice
  - For dynamic script files, SRI can typically not be enforced, so neither can Integrity Policy

- Reporting is a great way to gain insights into your application behavior
  - A report-only policy can make you aware of missing integrity attributes
  - Reporting on enforced policies can warn you about bugs or ongoing attacks

Integrity Policy ensures that all script loads must rely on SRI, a solid best practice for modern applications

In the future, Integrity Policy will also be able to make SRI mandatory for stylesheets

Integrity Policy also applies on dynamic script loads (by inserting script tags into the page at runtime)

How do you know what code is running?

SRI is great for static scripts, but does not always work without hiccups in modern applications

A new CSP feature allows us to monitor what code is loaded in the browser, giving us valuable insights

A sample Content Security Policy *header configuration with hash reporting enabled*

```
Reporting-Endpoints: csp-hashes="https://restograde.com/reports/hashes"
Content-Security-Policy-Report-Only:    script-src 'report-sha256';
                                        report-to csp-hashes
```

The CSP policy instructs the browser to send a hash for each file that is loaded to a specific endpoint

Reports are sent out-of-band, so it may take a while for them to show up

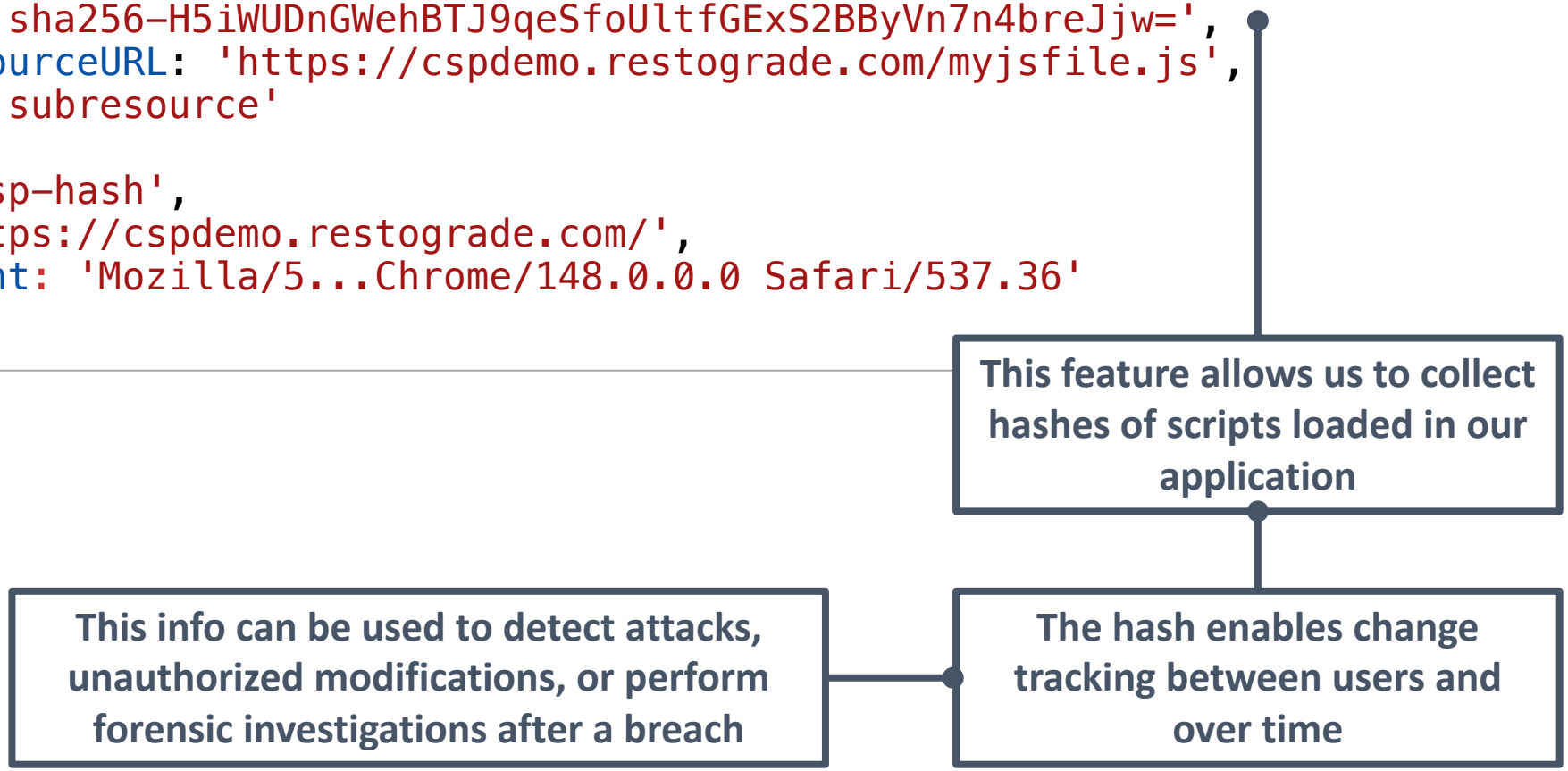
A sample CSP Hash *report*

```
{
  age: 36948,
  body: {
    destination: 'script',
    documentURL: 'https://cspdemo.restograde.com/',
    hash: 'sha256-H5iWUDnGWehBTJ9qeSfoUltfGExS2BByVn7n4breJjw=',
    subresourceURL: 'https://cspdemo.restograde.com/myjsfile.js',
    type: 'subresource'
  },
  type: 'csp-hash',
  url: 'https://cspdemo.restograde.com/',
  user_agent: 'Mozilla/5...Chrome/148.0.0.0 Safari/537.36'
}
```

The report contains all relevant information about the loaded code file as observed in the user's browser

A sample CSP Hash *report*

```
{
  age: 36948,
  body: {
    destination: 'script',
    documentURL: 'https://cspdemo.restograde.com/',
    hash: 'sha256-H5iWUDnGWehBTJ9qeSfoUltfGExS2BByVn7n4breJjw=',
    subresourceURL: 'https://cspdemo.restograde.com/myjsfile.js',
    type: 'subresource'
  },
  type: 'csp-hash',
  url: 'https://cspdemo.restograde.com/',
  user_agent: 'Mozilla/5...Chrome/148.0.0.0 Safari/537.36'
}
```



Report URI **Documentation**

Home Setup Integrations Help FAQ

Setup
Content Security Policy
CSP Wizard
CSP Integrity
Integrity Policy
Script Watch
Data Watch
Frame Watch
Policy Watch
Network Error Logging
Permissions Policy
Reporting API
Certificate Transparency
DMARC
Cross-Origin Embedder Policy
Cross-Origin Opener Policy
SMTP TLS
Sampling

Home › Setup

CSP Integ

What is CSP

CSP Integrity is a p
scripts that are run

Whether you have
and enable CSP Int
running on your pa

Enable Integ

If you already have
`sha256'` keyword t

```
script-src examp
```

If you do not have
keyword:

```
Content-Security
```

CSP Integrity Reports

Page 1 of 10 | View 100 records

Filter Export

Date	URI	Destination	Subresource	Raw	Count
Days 09 Sept 2025	report-uri.com /{agg}	destination	subresource hostname subresource path subresource hash		All All
09 Sep 2025 14:54:54	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/js/refresh/frontend.min.js sha256-3blw5/58mQ… unknown	show	220
09 Sep 2025 14:53:47	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/libs/refresh/bootstrap/bootstrap.bundle.min.js sha256-wMCQIK229gKxbUg3QWa544ypI4OoFIC2qQI8Q8xD8x8= from bootstrap	show	217
09 Sep 2025 14:38:16	https://report-uri.com/	script	https://cdn.report-uri.com/js/refresh/home-payment-switcher.min.js sha256-t24ohdedLQ… unknown	show	214
09 Sep 2025 14:54:54	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/libs/jquery/3.5.1/jquery.min.js sha256-9/aliU8dGd… from jquery	show	114
09 Sep 2025 14:53:33	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/libs/bootstrap-switch/3.3.2/js/bootstrap-switch.min.js sha256-sAkXLwDFSA… from bootstrap-switch	show	112
09 Sep 2025 14:54:54	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/libs/jquery-cookie/1.4.1/jquery.cookie.min.js sha256-1A78rJEdiW… from jquery-cookie	show	112
09 Sep 2025 14:47:24	https://report-uri.com/{agg}	script	https://cdn.report-uri.com/libs/jquery-migrate/3.3.2/jquery-migrate.min.js sha256-EG/NjXl+2n… from jquery-migrate	show	110

Want to try this out? Use the *PWS* discount code for 25% off

https://docs.report-uri.com/setup/csp-integrity/

Hash reporting is brand new but exciting. Keep an eye on how this feature will be used and evolve!

The web is a complicated environment, with lots of hard-to-control aspects that negatively impact security

Browsers have started offering more and better security tools, giving us unprecedented insights and control

Key takeaways

1. Leverage the core security model of the web for baseline security
2. Cookies are good! Security flags and prefixes make them even better
3. New JavaScript security mechanisms can help with the scriptocalypse

Thank you!

Need training or security guidance?
Reach out to discuss how I can help

https://pragmaticwebsecurity.com