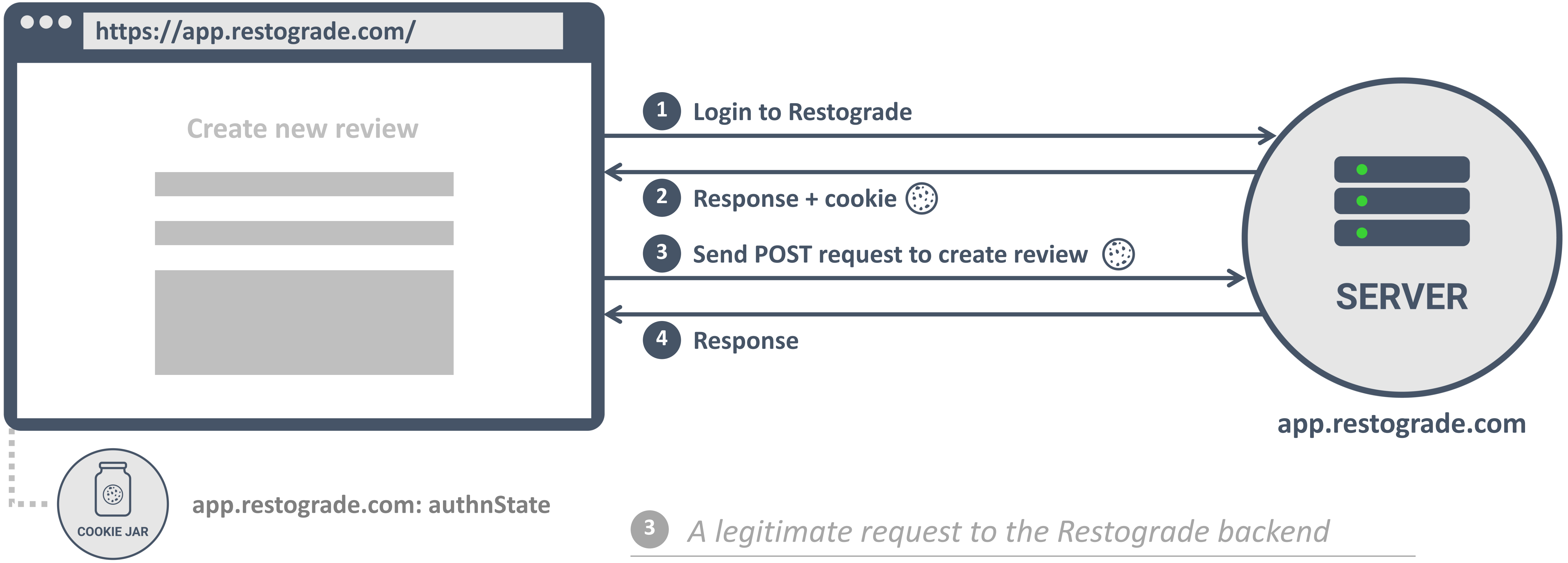
The past, present, and future of Cross-Site Request Forgery

Dr. Philippe De Ryck

https://Pragmatic Web Security.com

WTF is CSRF?

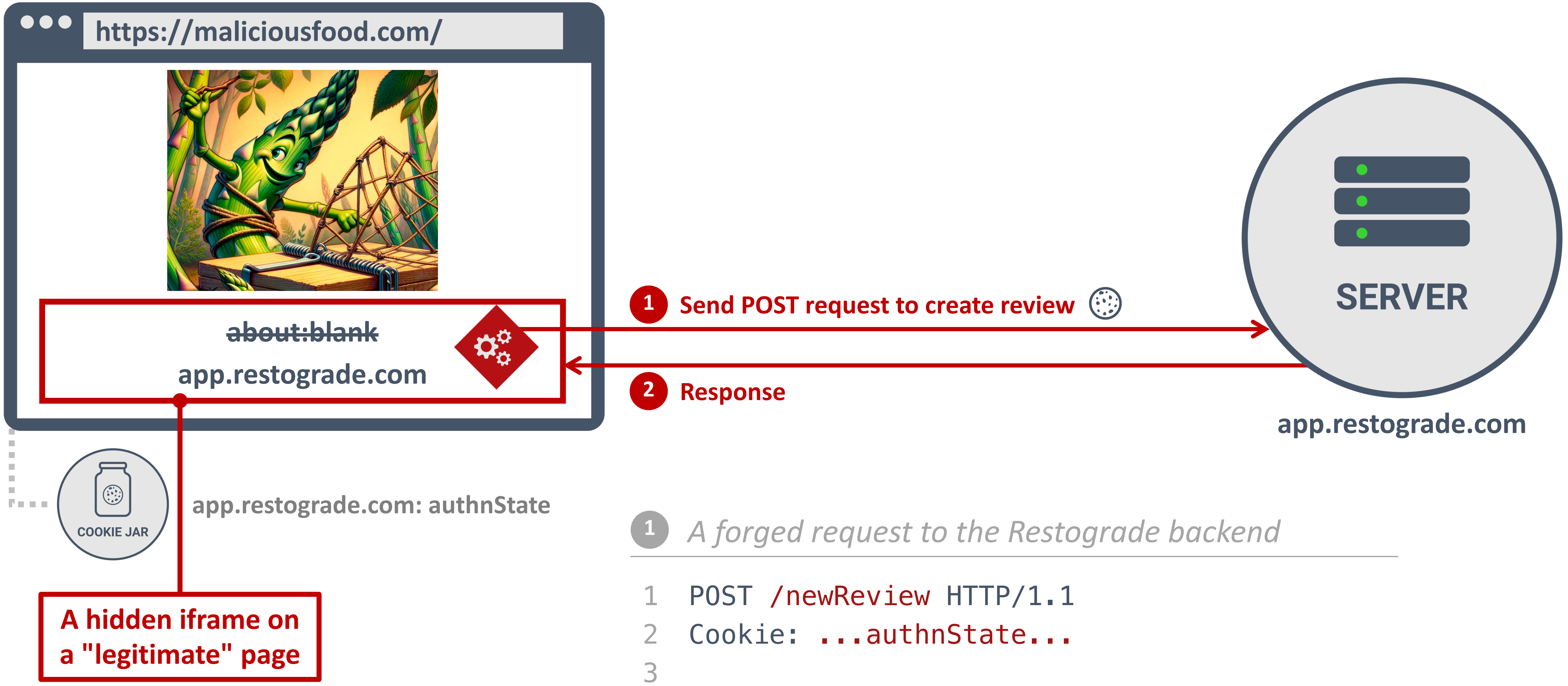
Setting the scene for Cross-Site Request Forgery (CSRF)



3 *A legitimate request to the Restograde backend*

```
POST /newReview HTTP/1.1
Cookie: ...authnState...

restaurant=1&title=…&content=…
```

A FORM-BASED CSRF ATTACK


https://maliciousfood.com/
~~about:blank~~
app.restograde.com
1 Send POST request to create review
2 Response
SERVER
app.restograde.com
COOKIE JAR
app.restograde.com: authnState
A hidden iframe on a "legitimate" page

1 *A forged request to the Restograde backend*

```
POST /newReview HTTP/1.1
Cookie: ...authnState...

restaurant=1&title=…&content=…
```

Traditional CSRF in action

CSRF ATTACKS AFFECT TRADITIONAL SERVER-SIDE APPS

CSRF attacks exist because the browser automatically attaches cookies to outgoing requests.

CSRF used to be a real problem for traditional server-side applications

I am *Dr. Philippe De Ryck*

Pragmatic Web Security
Security for developers

Founder of Pragmatic Web Security

Google Developers Experts

Google Developer Expert

AMBASSADOR PROGRAM

Auth0 Ambassador

SecAppDev

SecAppDev organizer

I help developers with security

✓ Hands-on in-depth security training

✓ Advanced online security courses

✓ Security advisory services

https://pdr.online

GRAB A COPY OF THE SLIDES ...

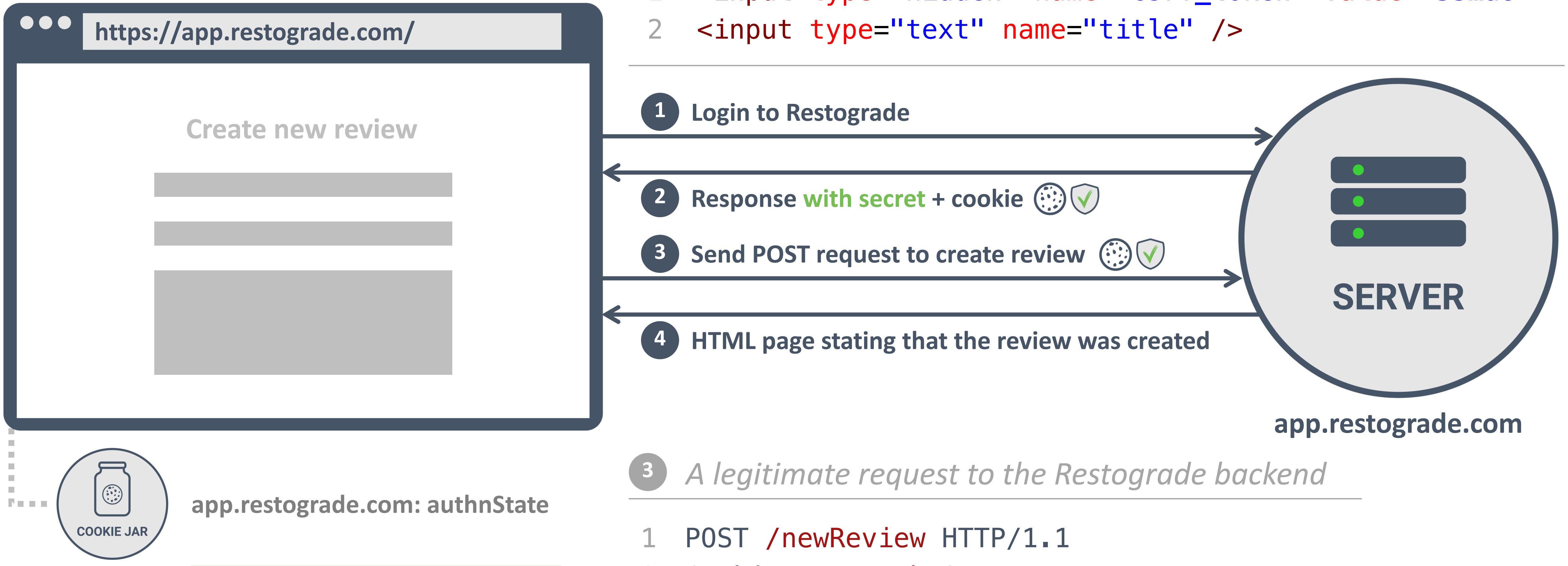
https://pragmaticwebsecurity.com/talks

/in/PhilippeDeRyck

https://infosec.exchange/@PhilippeDeRyck

How do we stop CSRF attacks?

CSRF DEFENSE: SYNCHRONIZER TOKENS



https://app.restograde.com/
Create new review
1 Login to Restograde
2 Response with secret + cookie
3 Send POST request to create review
4 HTML page stating that the review was created
SERVER
app.restograde.com
COOKIE JAR
app.restograde.com: authnState

② *A CSRF token in a hidden form field*

```html
<input type="hidden" name="csrf_token" value="53…a8">
<input type="text" name="title" />
```

③ *A legitimate request to the Restograde backend*

```
1 POST /newReview HTTP/1.1
2 Cookie: ...authnState...
3
4 restaurant=1&title=…&csrf_token=530…ea8
```

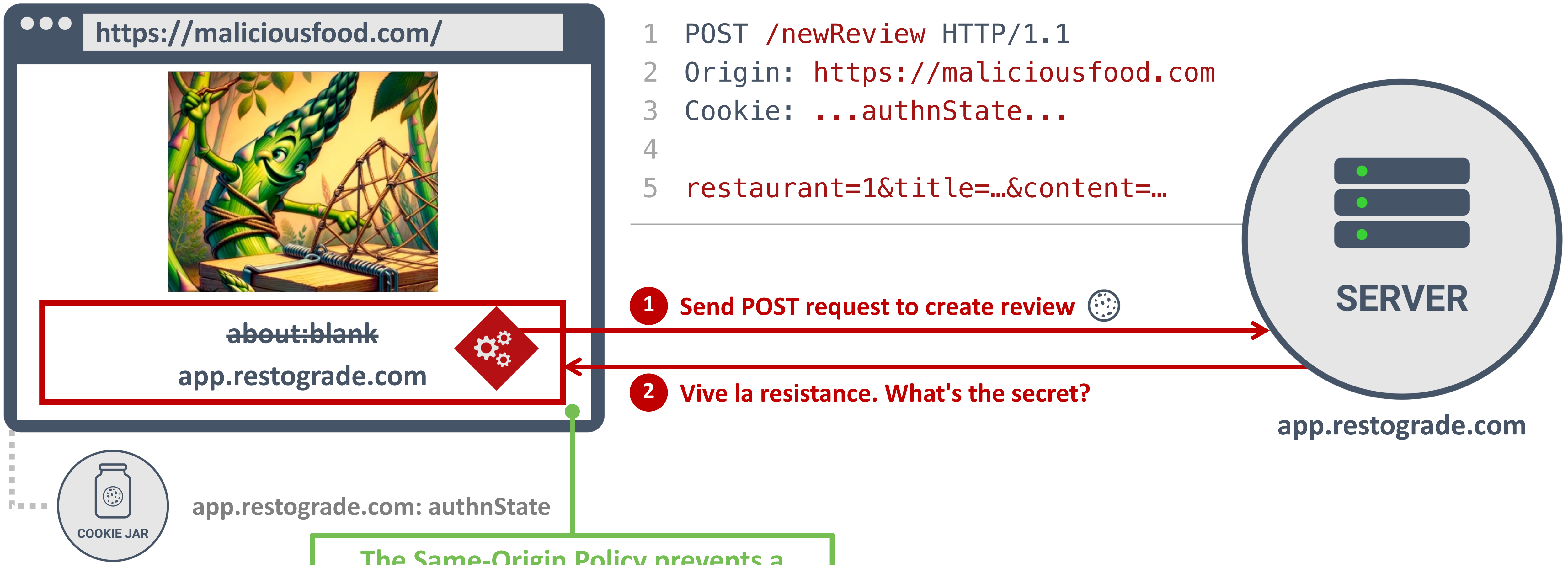
The hidden CSRF token is submitted as part of the form data

pdr.online

CSRF DEFENSE: SYNCHRONIZER TOKENS

1 *A forged request to the Restograde backend*

```
POST /newReview HTTP/1.1
Origin: https://maliciousfood.com
Cookie: ...authnState...

restaurant=1&title=…&content=…
```



https://maliciousfood.com/
~~about:blank~~
app.restograde.com
1 Send POST request to create review
2 Vive la resistance. What's the secret?
SERVER
app.restograde.com
COOKIE JAR
app.restograde.com: authnState

The Same-Origin Policy prevents a malicious page from stealing a legitimate token from a page from app.restograde.com

Synchronizer tokens are a good CSRF defense

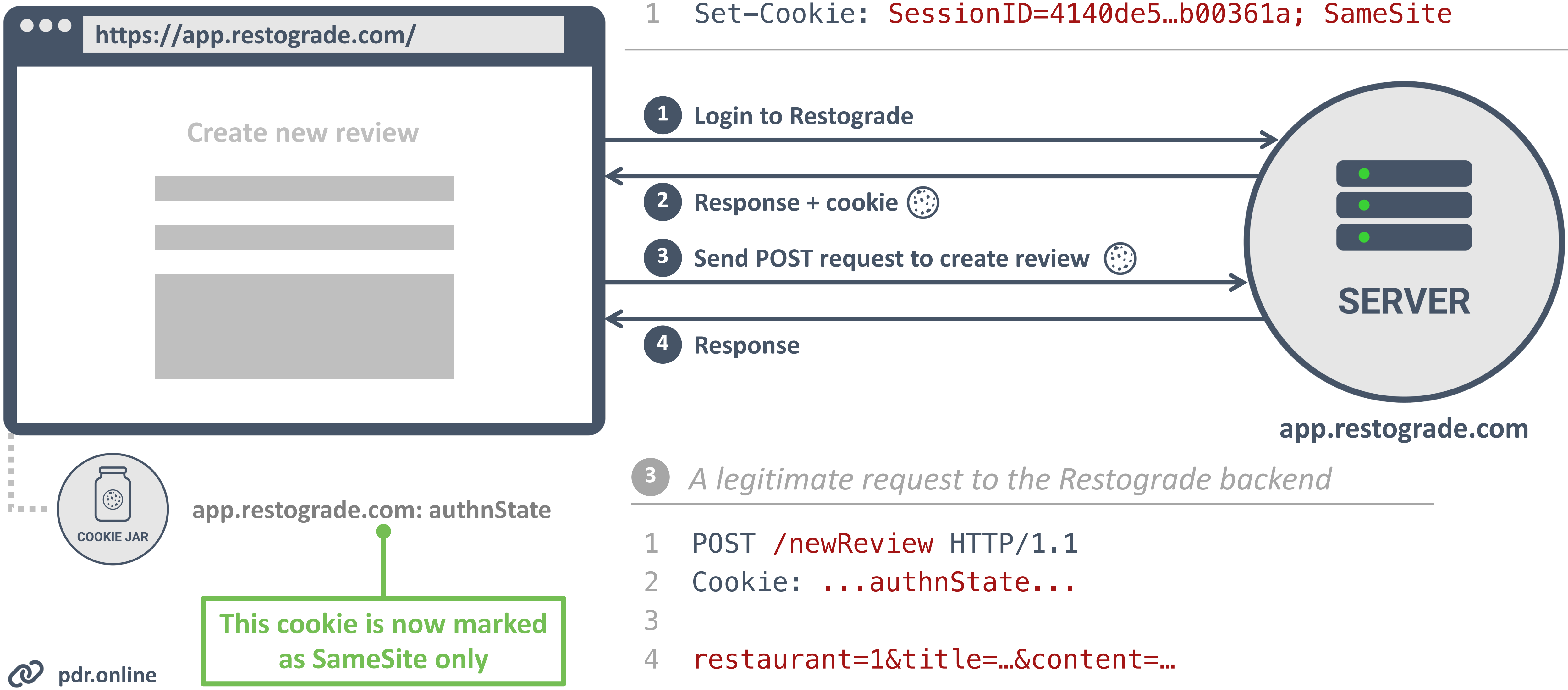
By requiring the browser to submit a secret token along with the request data, the backend can identify and reject illegitimate requests.

The use of synchronizer tokens requires explicit implementation effort and is often forgotten or omitted

ChatGPT

Here's an illustration representing the concept of 'SameSite' cookies in the context of internet browsing.

CSRF defense: SameSite cookies


https://app.restograde.com/
Create new review
1 Login to Restograde
2 Response + cookie
3 Send POST request to create review
4 Response
SERVER
app.restograde.com
COOKIE JAR
app.restograde.com: authnState
This cookie is now marked as SameSite only

2 Setting a SameSite cookie

```
Set-Cookie: SessionID=4140de5…b00361a; SameSite
```

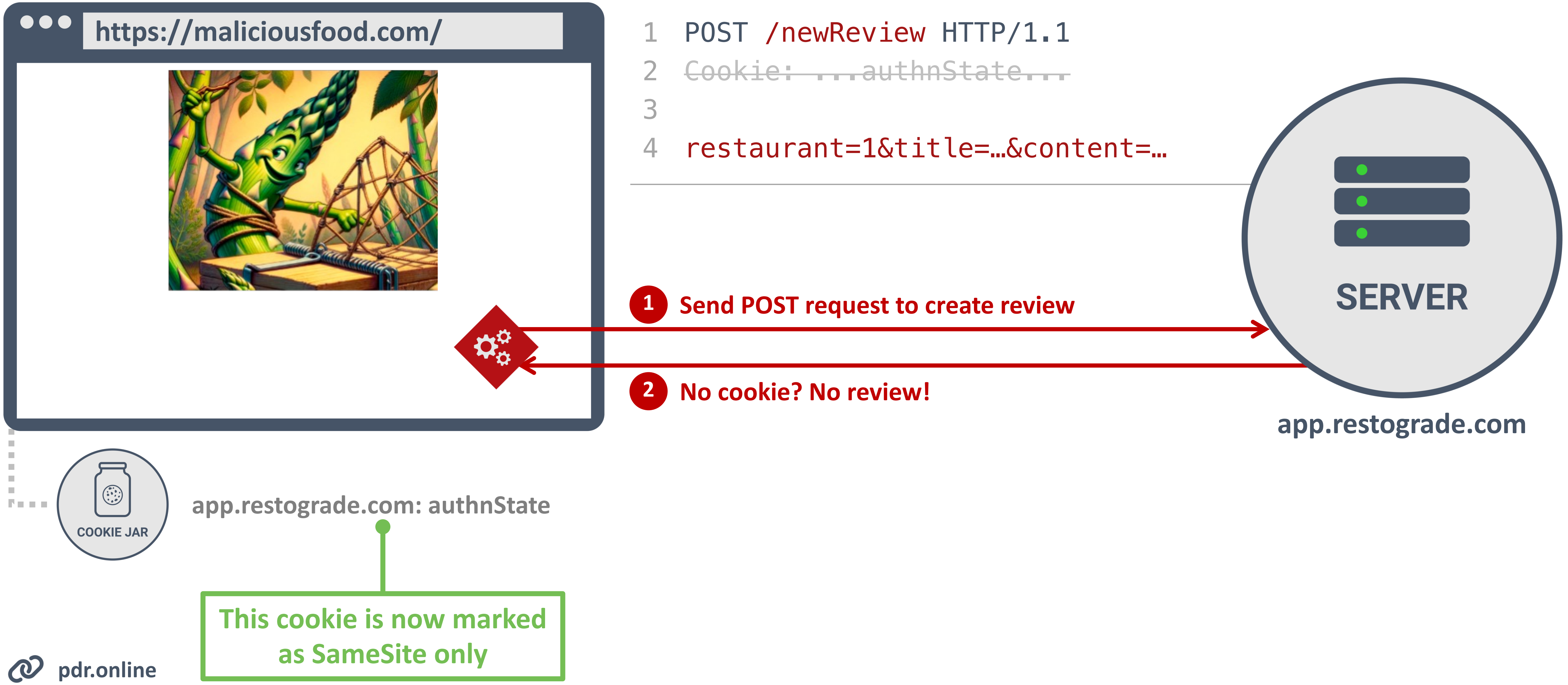
3 A legitimate request to the Restograde backend

```
POST /newReview HTTP/1.1
Cookie: ...authnState...

restaurant=1&title=…&content=…
```

CSRF defense: SameSite cookies

1 *A forged request to the Restograde backend*

```
POST /newReview HTTP/1.1
~~Cookie: ...authnState...~~

restaurant=1&title=…&content=…
```


https://maliciousfood.com/
1 Send POST request to create review
2 No cookie? No review!
SERVER
app.restograde.com
COOKIE JAR
app.restograde.com: authnState
This cookie is now marked as SameSite only

SameSite cookies neutralize CSRF

SameSite cookies are not included on cross-site requests.

The attacker can still send the request, but cookie-based authentication state will not be included by the browser.

Aug 11, 2020

GOOGLE ROLLS OUT SAMESITE COOKIE CHANGES TO CHROME

By Fahmida Y. Rashid

Share

pdr.online

https://duo.com/decipher/google-rolls-out-samesite-cookie-changes-to-chrome

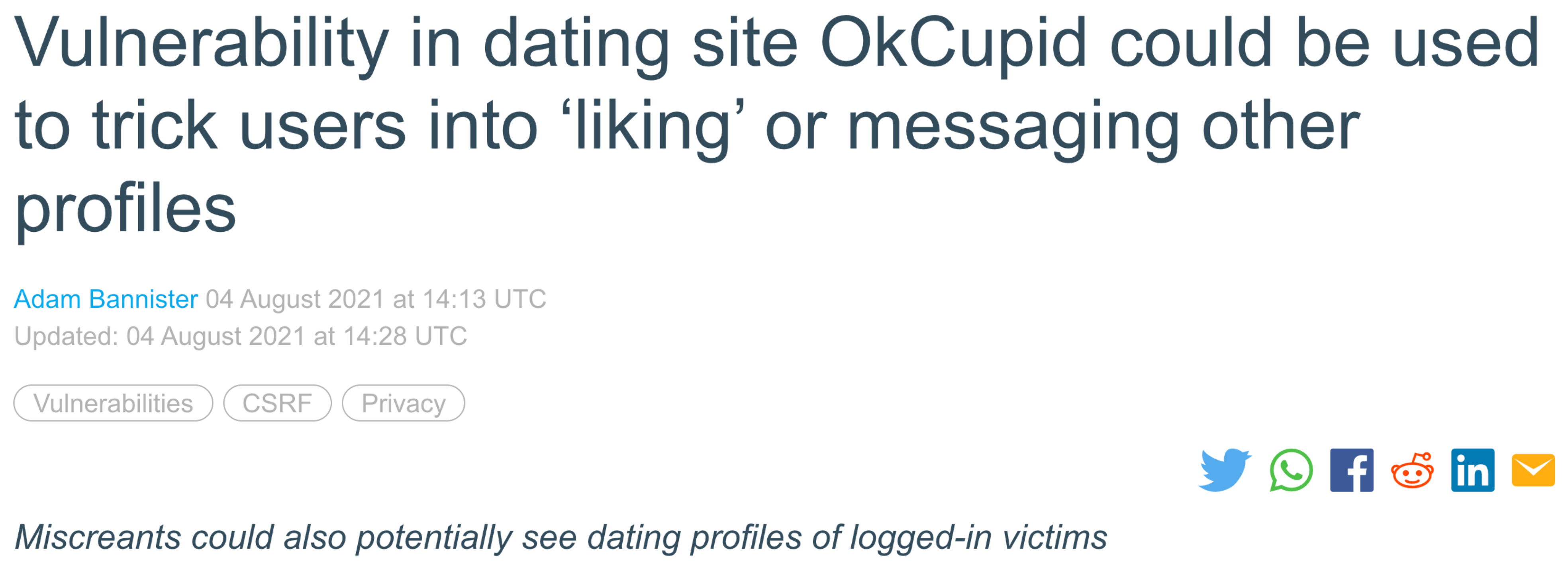
SameSite cookies in action

Chrome treats cookies as SameSite by default

Since 2020, Chrome treats cookies as SameSite, unless they set SameSite=None.

Note that other browsers do not, so you still need to set the SameSite flag to mitigate CSRF attacks.

What about APIs?



Vulnerability in dating site OkCupid could be used to trick users into 'liking' or messaging other profiles

Adam Bannister 04 August 2021 at 14:13 UTC
Updated: 04 August 2021 at 14:28 UTC

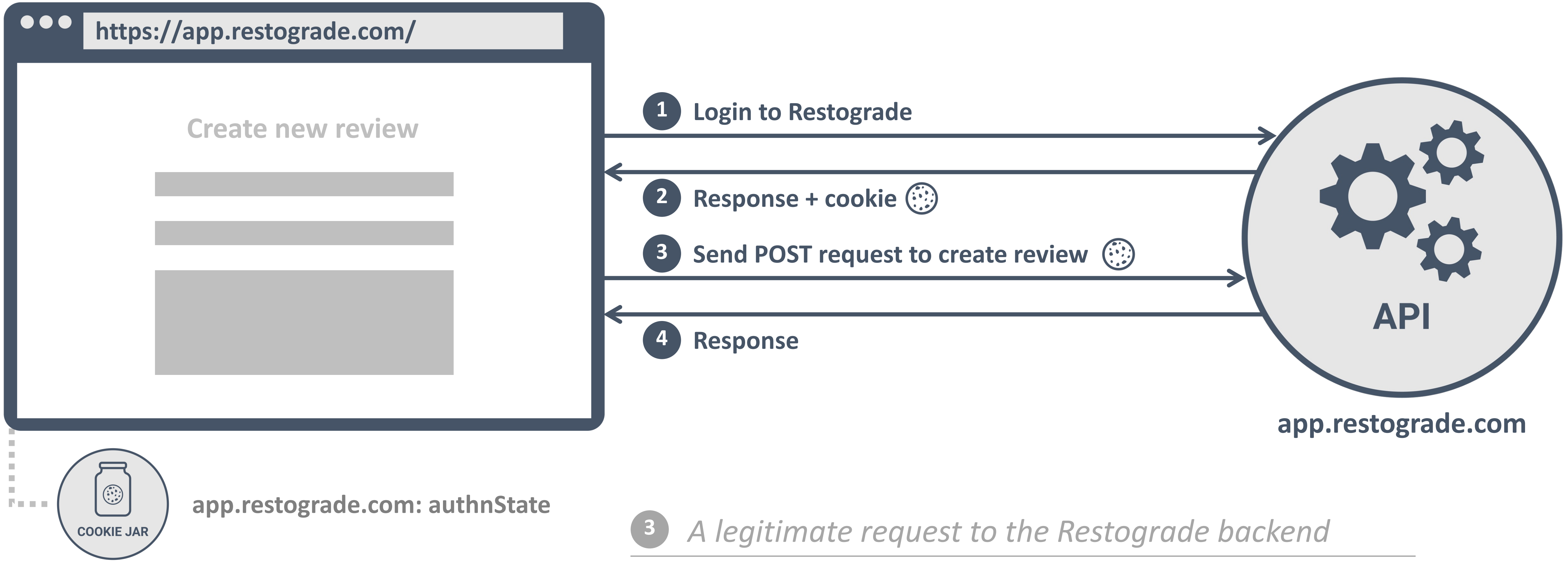
Vulnerabilities CSRF Privacy

Miscreants could also potentially see dating profiles of logged-in victims

https://portswigger.net/daily-swig/vulnerability-in-dating-site-okcupid-could-be-used-to-trick-users-into-liking-or-messaging-other-profiles

“ **Zhu also investigated whether other sites' authenticated endpoints similarly accepted POSTs with content-type: text/plain, despite expecting JSON.** ”

Setting the scene for Cross-Site Request Forgery (CSRF)

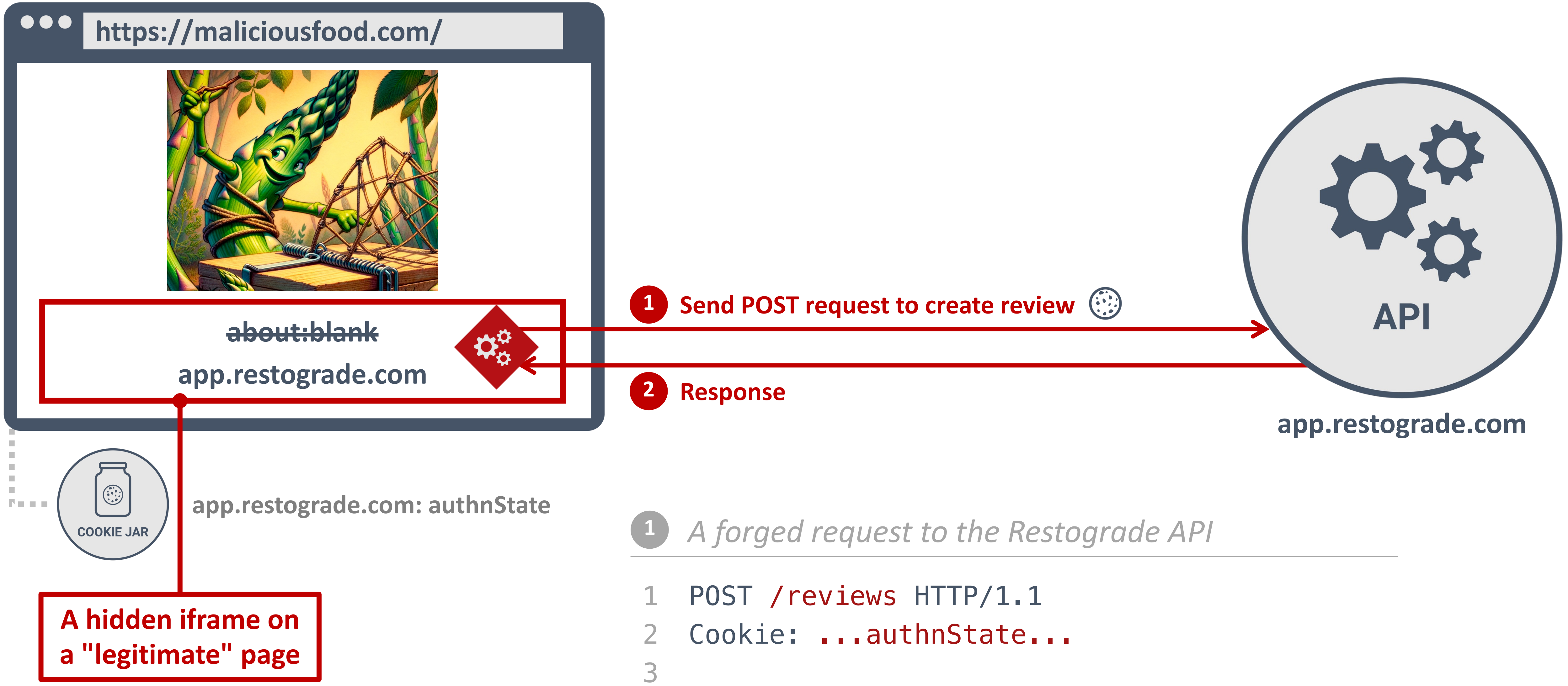


3 *A legitimate request to the Restograde backend*

```
POST /reviews HTTP/1.1
Cookie: ...authnState...

{"restaurant":1,"title":"…","content":"…"}
```

pdr.online

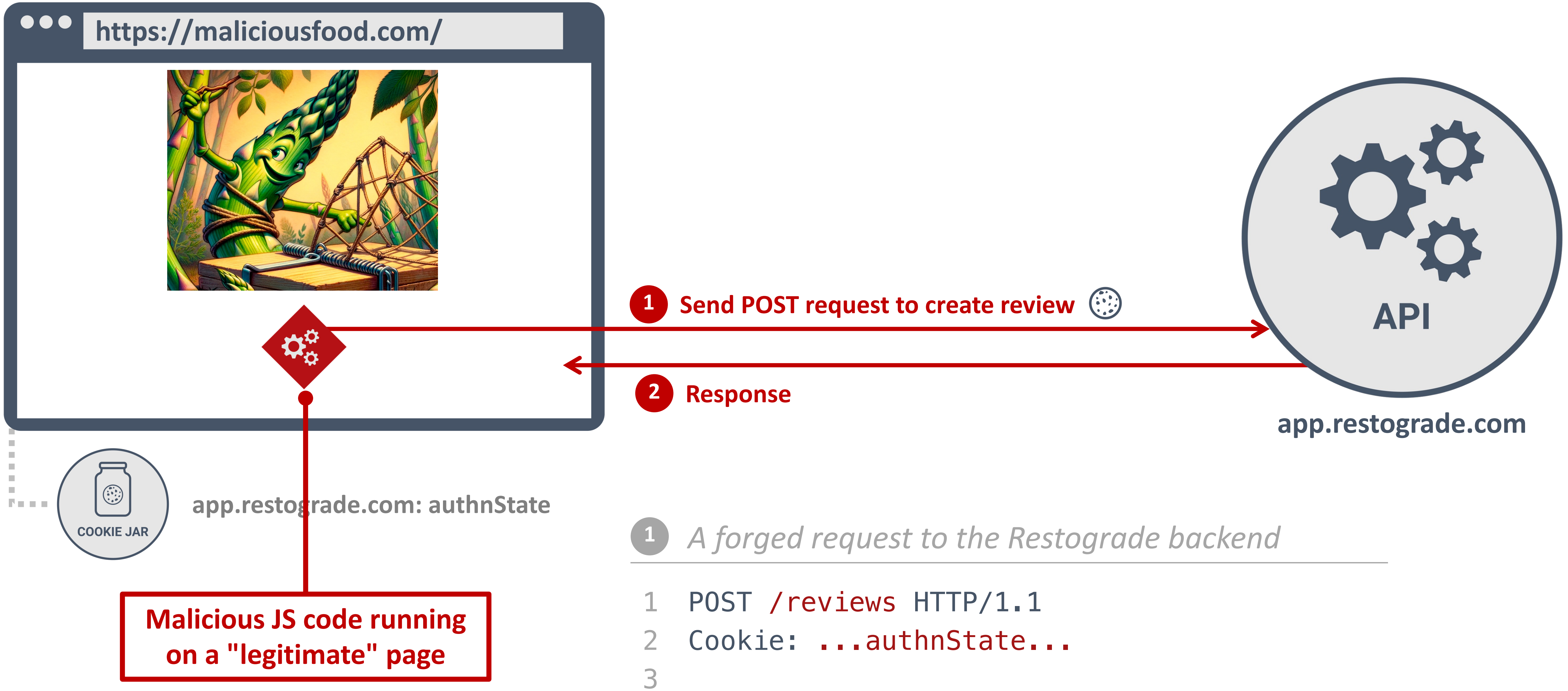
A FORM-BASED CSRF ATTACK


https://maliciousfood.com/
~~about:blank~~
app.restograde.com
1 Send POST request to create review
2 Response
API
app.restograde.com
COOKIE JAR
app.restograde.com: authnState
A hidden iframe on a "legitimate" page

1 *A forged request to the Restograde API*

```
POST /reviews HTTP/1.1
Cookie: ...authnState...

{"restaurant":1,"title":"…","content":"…"}
```

A FETCH-BASED CSRF ATTACK


https://maliciousfood.com/
1 Send POST request to create review
2 Response
API
app.restograde.com
COOKIE JAR
app.restograde.com: authnState
Malicious JS code running on a "legitimate" page

1 *A forged request to the Restograde backend*

```
POST /reviews HTTP/1.1
Cookie: ...authnState...

{"restaurant":1,"title":"…","content":"…"}
```

Abusing APIs with CSRF

SameSite cookies also prevent CSRF against APIs

Cookie-based APIs need to worry about CSRF

APIs that rely on cookies are less common, but they definitely exist (e.g., OAuth BFFs).

APIs relying on cookies need to ensure they properly mitigate CSRF attacks.

pdr.online

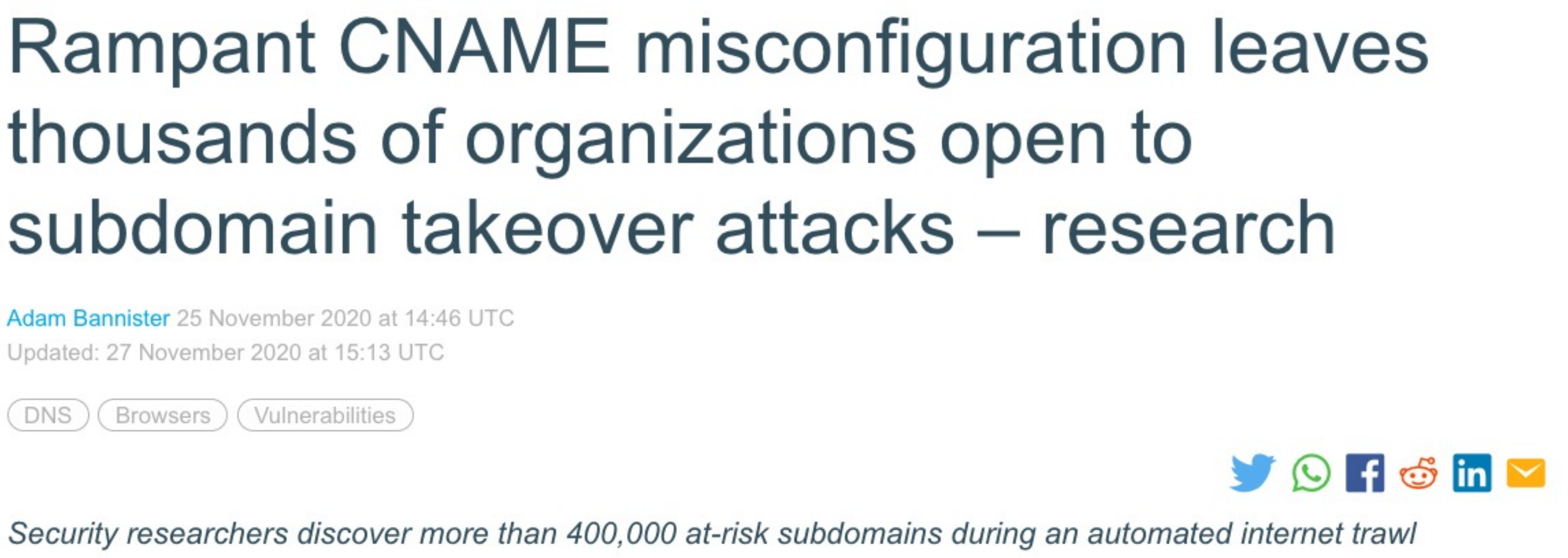
SameSite cookies effectively mitigate
Cross-Site Request Forgery attacks

SameSite cookies cannot protect against
Cross-Origin (but Same-Site) Request Forgery

From Cross-Site to Cross-Origin Request Forgery

Why would we ever give an attacker control over a subdomain?

SOME MEN
JUST WANT TO WATCH THE WORLD BURN.


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

Adam Bannister 25 November 2020 at 14:46 UTC
Updated: 27 November 2020 at 15:13 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

“

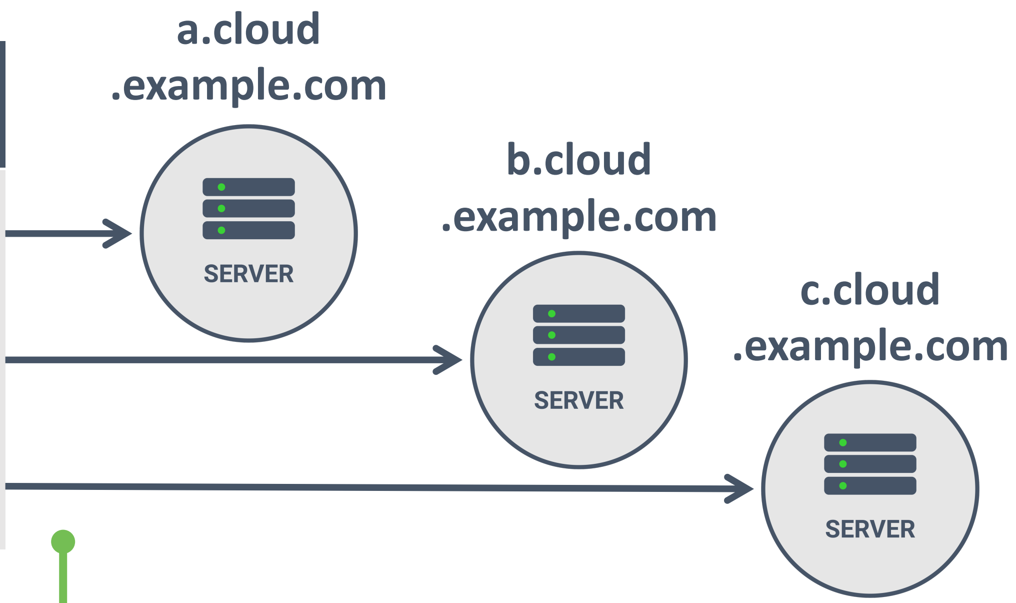
Attackers can serve malicious content to hijack user's sessions by abusing OAuth 2.0 redirect URIs

”

LOSING CONTROL OF A SUBDOMAIN

Restograde DNS records

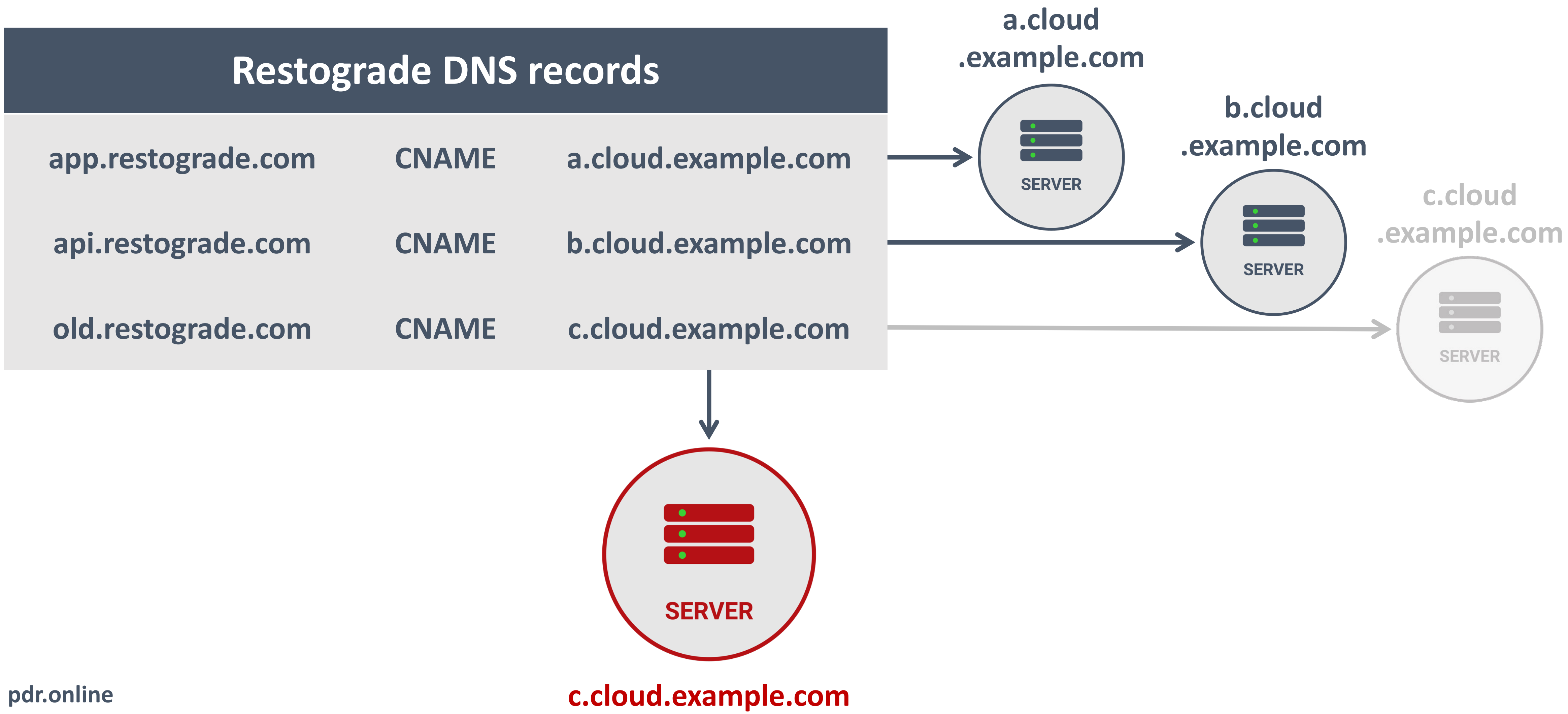
app.restograde.com	CNAME	a.cloud.example.com
api.restograde.com	CNAME	b.cloud.example.com
old.restograde.com	CNAME	c.cloud.example.com



CNAME-based configurations are common in cloud deployments.

There is nothing wrong with this configuration as long as all cloud servers remain active

LOSING CONTROL OF A SUBDOMAIN

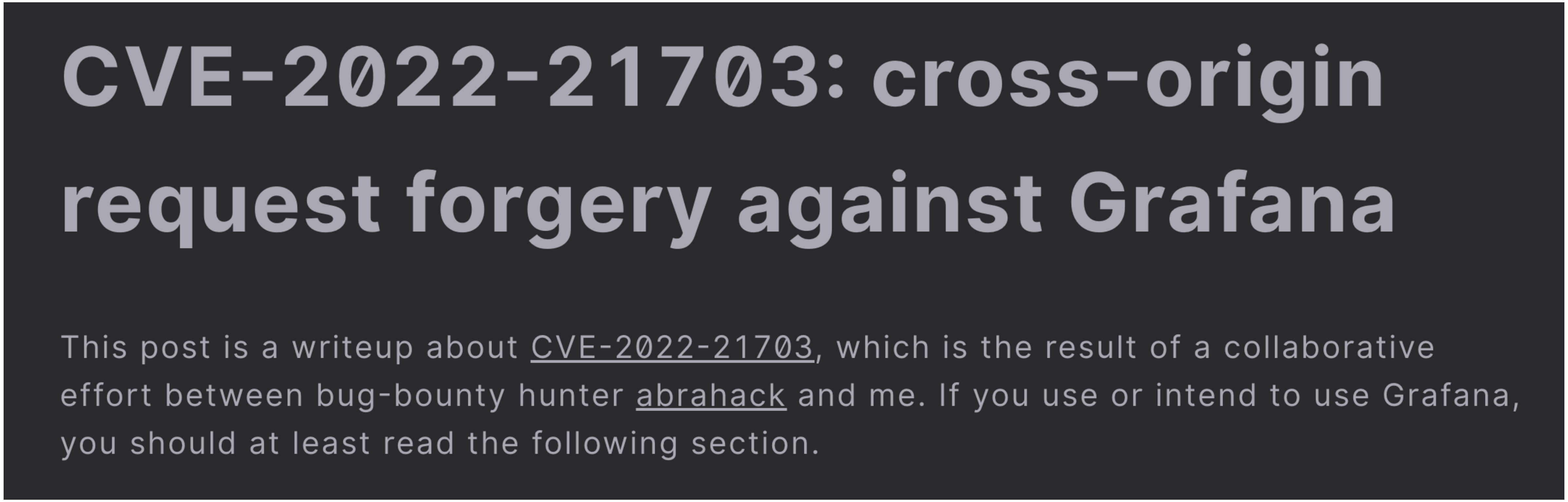


Restograde DNS records		
app.restograde.com	CNAME	a.cloud.example.com
api.restograde.com	CNAME	b.cloud.example.com
old.restograde.com	CNAME	c.cloud.example.com

CSRF IS DEAD, LONG LIVE CORF!

While Cross-Site Request Forgery may be on the way out, Cross-Origin (but same-site) Request Forgery is definitely gaining traction.

Please tell me you're making this up?


CVE-2022-21703: cross-origin request forgery against Grafana

This post is a writeup about CVE-2022-21703, which is the result of a collaborative effort between bug-bounty hunter abrahack and me. If you use or intend to use Grafana, you should at least read the following section.

https://jub0bs.com/posts/2022-02-08-cve-2022-21703-writeup/

“All GET- and POST-based endpoints of Grafana’s HTTP API are affected.”

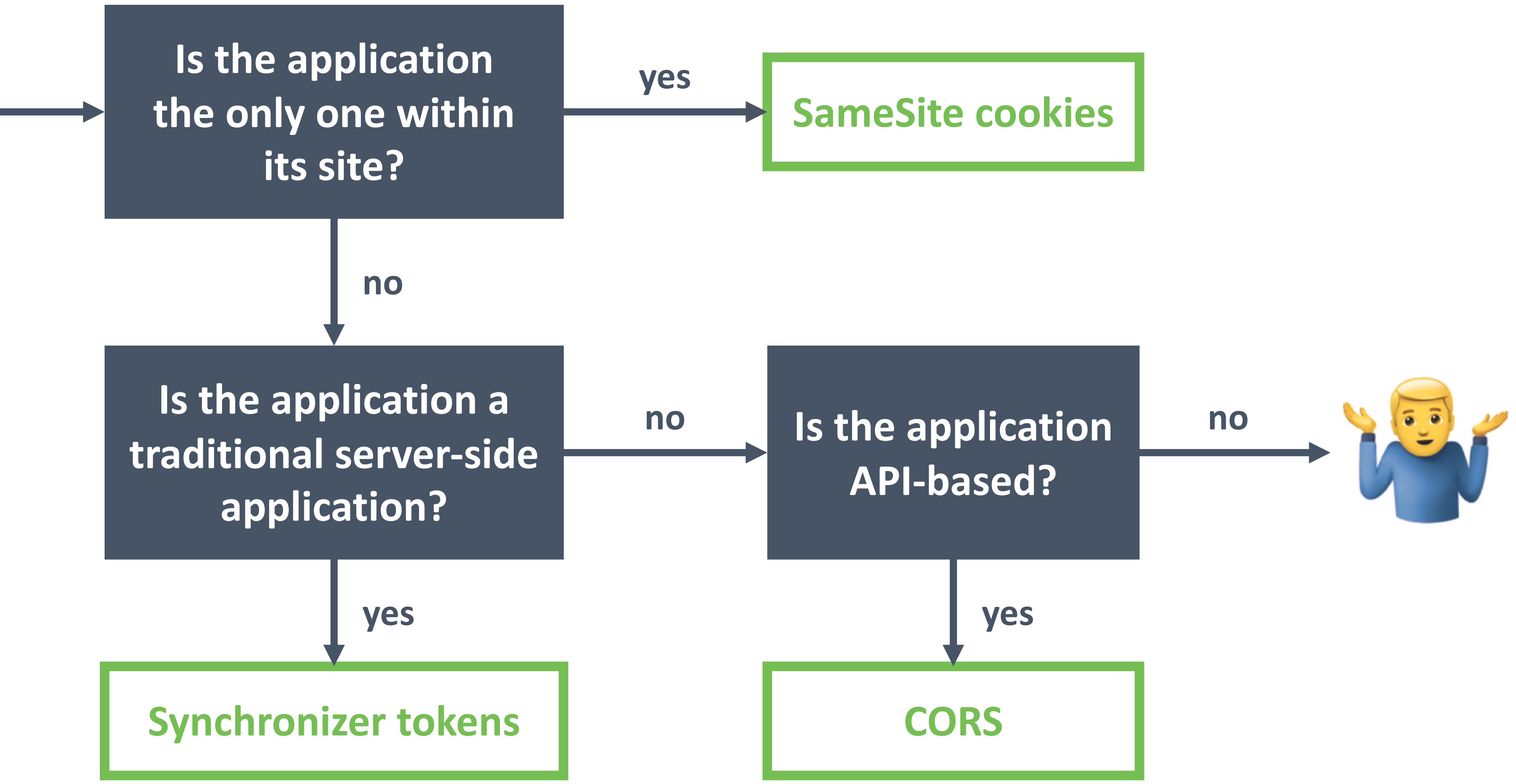
Why does that even work?

Deep-dive into CSRF in APIs

APIs can rely on CORS as a CSRF defense

Cookie-based APIs accepting non-CORS-safelisted requests are subject to Cross-* Request Forgery.

APIs should restrict HTTP methods and content types, and force the use of CORS requests by requiring the client to include a custom request header.


Is the application the only one within its site?
yes
SameSite cookies
no
Is the application a traditional server-side application?
no
Is the application API-based?
no
yes
Synchronizer tokens
yes
CORS

KEY TAKEAWAYS

1. CSRF matters when you rely on cookies for user authN/authZ
2. SameSite cookies mitigate CSRF, but not Cross-Origin Request Forgery
3. APIs can rely on CORS as a defense against Cross * Request Forgery

Thank you!

Reach out to discuss
how I can help you with security

https://pragmaticwebsecurity.com