

## 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.

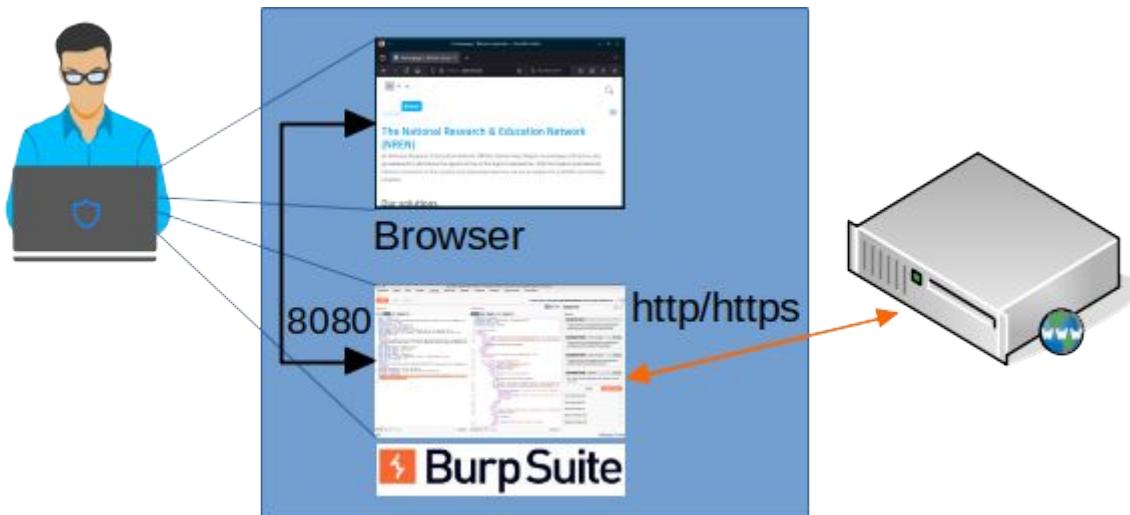
Burp Suite is currently available in three editions:

- 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.

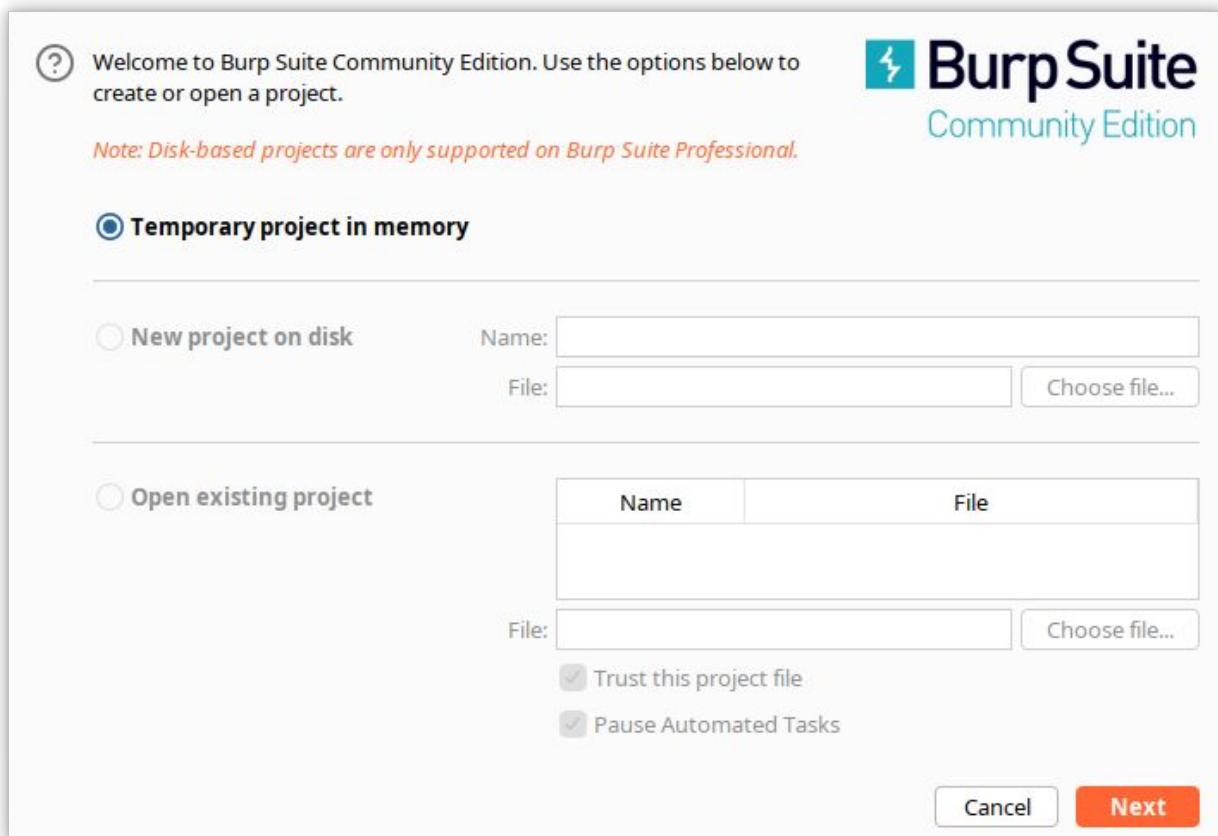
## Monitoring web traffic

The following description is based on the free Community Edition.

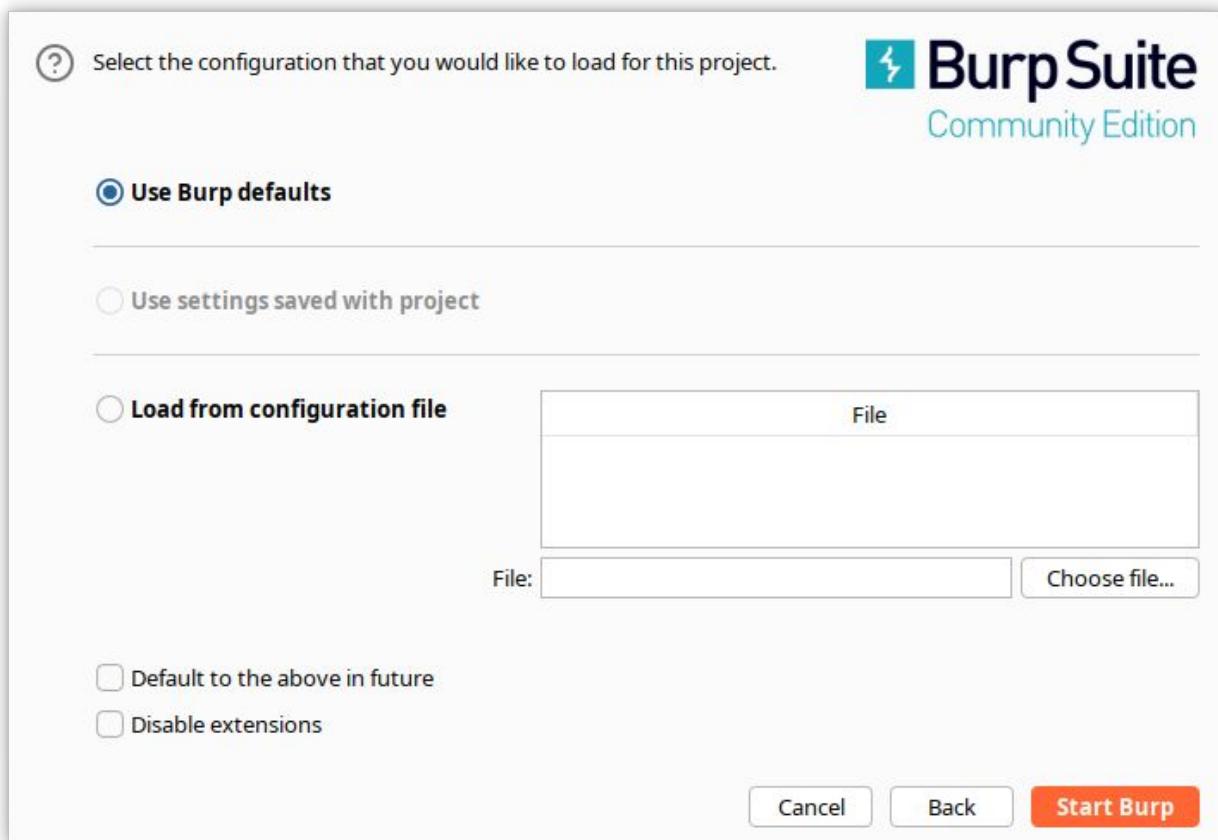
### Starting Burp Suite

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.

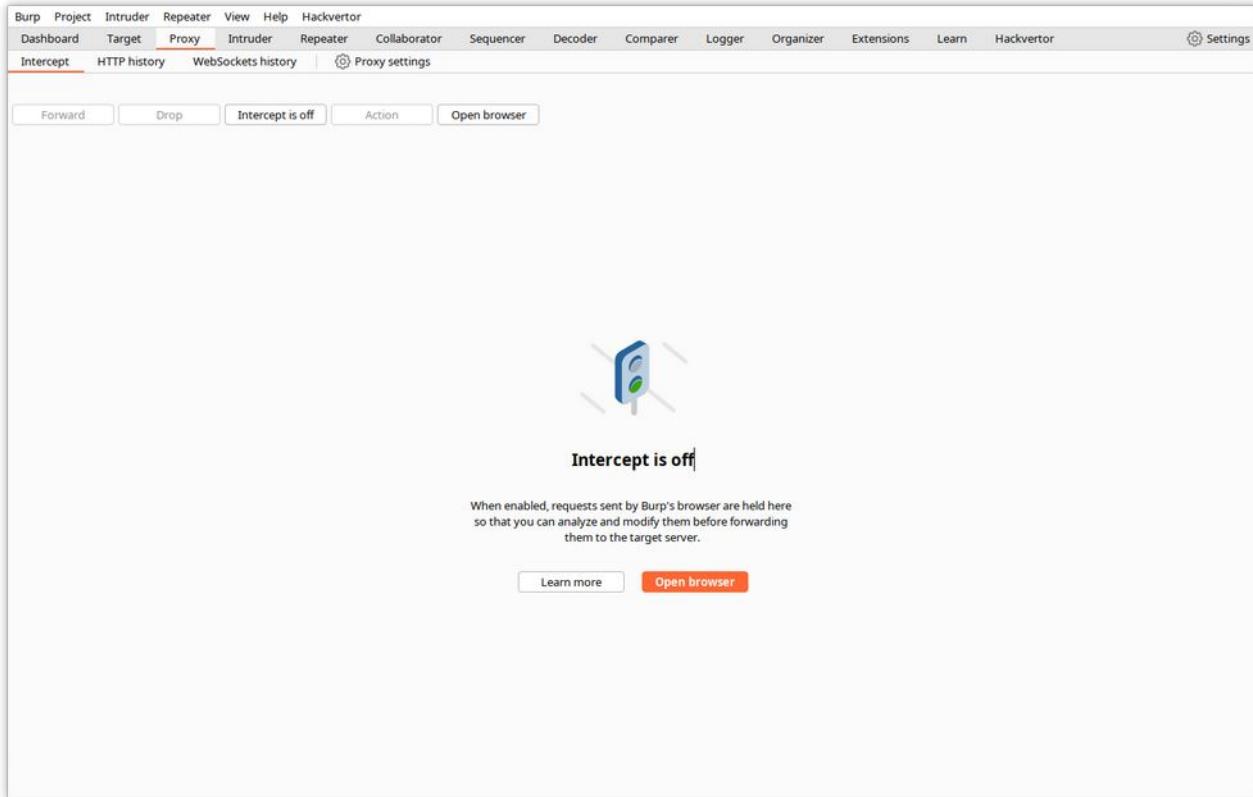


Click **Start Burp**. The *Dashboard* screen opens.

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... /
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.

The left screenshot shows the Burp Suite interface with the 'Intercept' tab selected. A central message states 'Intercept is off' with a description: 'When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.' Below the message are 'Learn more' and 'Open browser' buttons. The right screenshot shows a separate browser window displaying the Portswigger website, specifically the 'STATE MACHINE' research page. The page features a large blue banner with the text 'STATE MACHINE' and several call-to-action buttons: 'Discover the latest research →', 'Try the techniques for yourself →', and 'Get the latest product capabilities →'. At the bottom, there are sections for 'Web Security Academy', 'Documentation and support', and 'Join the community'.

## 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' tab is active, displaying two captured requests:

- Request 4: POST / (Host: localhost:4321, Method: POST, URL: /, Status code: 200, Length: 238, MIME type: HTML, Extension: HTML, Title: , Notes: , TLS: 127.0.0.1, IP: 127.0.0.1)
- Request 3: GET / (Host: localhost:4321, Method: GET, URL: /, Status code: 200, Length: 521, MIME type: HTML, Extension: HTML, Title: , Notes: , TLS: 127.0.0.1, IP: 127.0.0.1)

The 'Request' and 'Response' panes show the raw data. The 'Request' pane for the POST request includes headers like Host, Cache-Control, User-Agent, Accept, and Sec-Fetch-Site. The 'Response' pane for the GET request shows the HTML content of a login form.

The 'Inspector' pane on the right provides detailed information about the selected request, including Request attributes (2), Request headers (15), Response headers (3), and Notes.

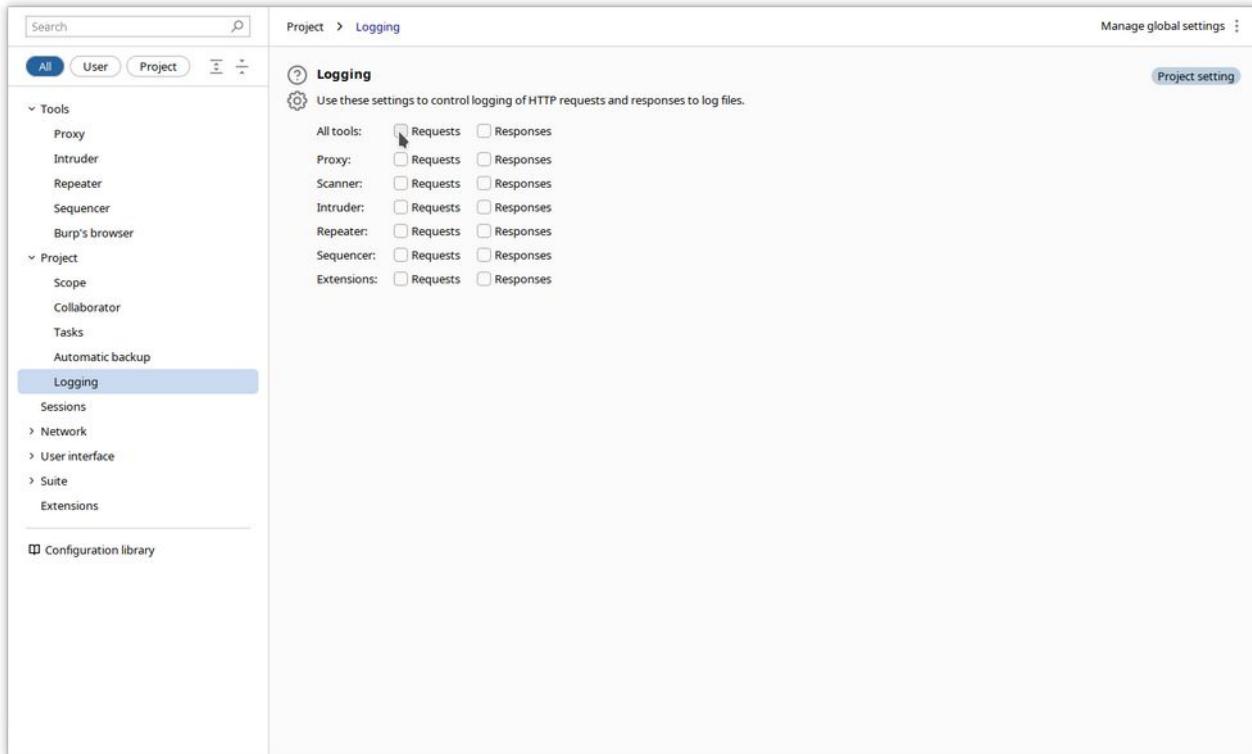
The captured data for the request contain the HTTP request type (GET), the requested path (/) and the complete headers (Host, Cache-Control, User-Agent, accepted document types...), the cookies and the variable values (for POST requests).

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

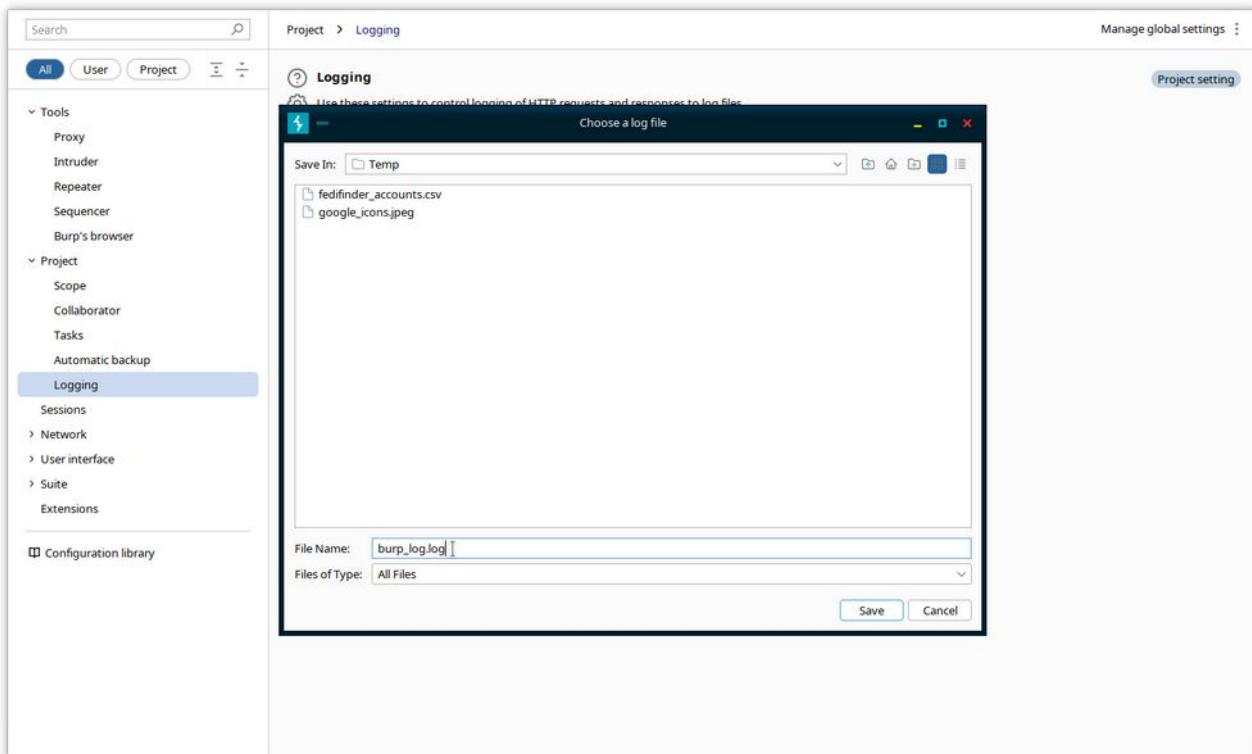
## Logging traffic

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 left of the Burp window, then in the left column, click on **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->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

=====

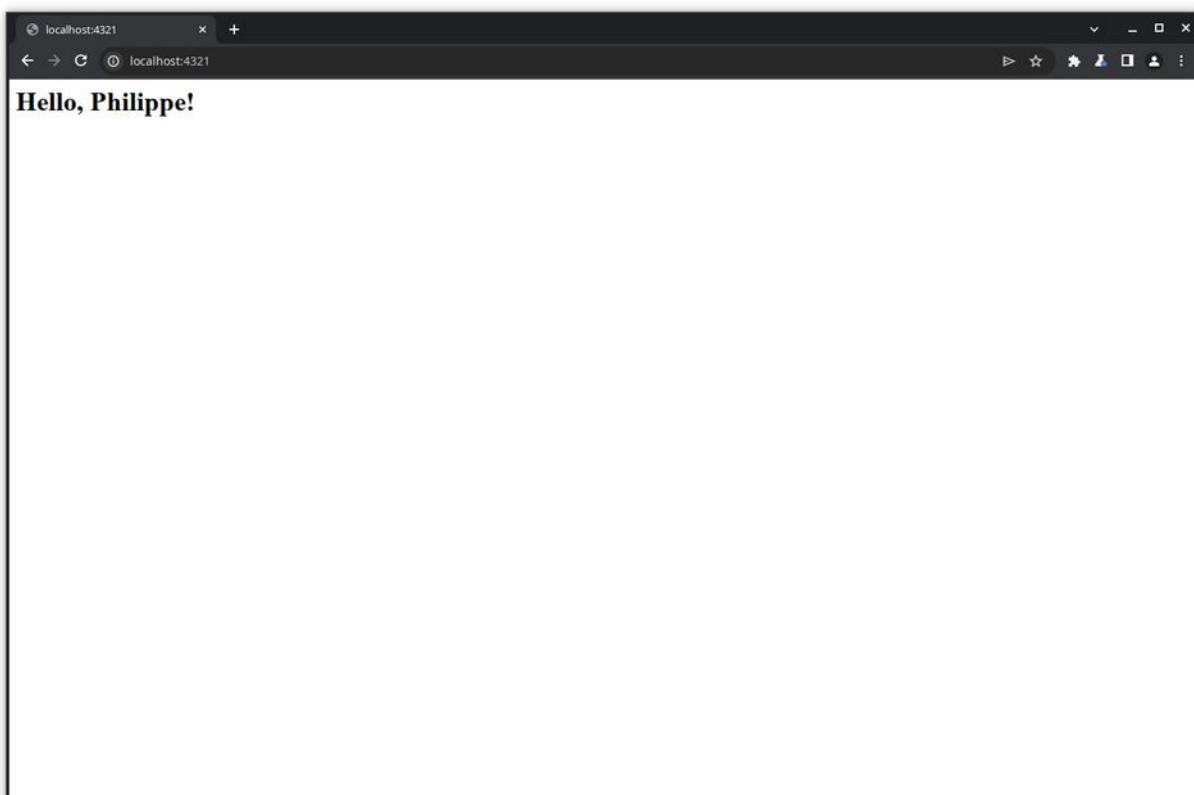
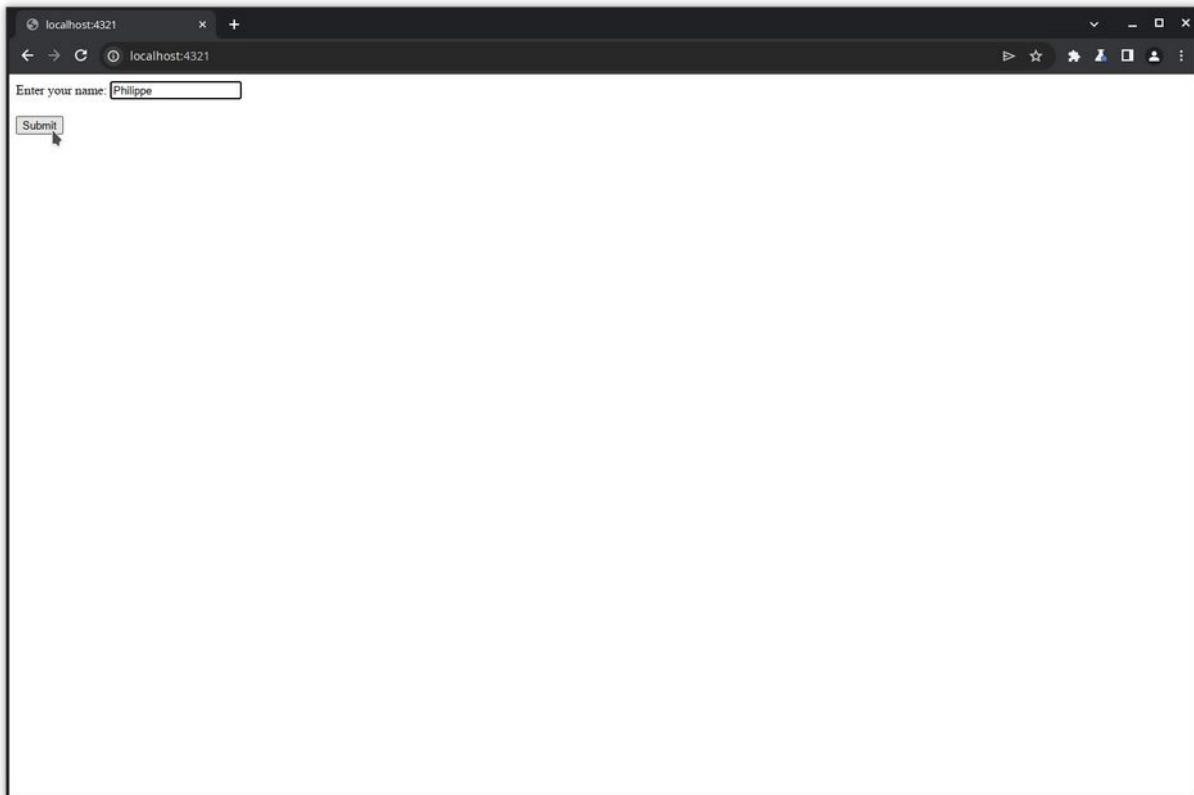
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.8.10
Date: Tue, 01 Oct 2024 17:03:15 GMT
Content-Type: text/html

<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

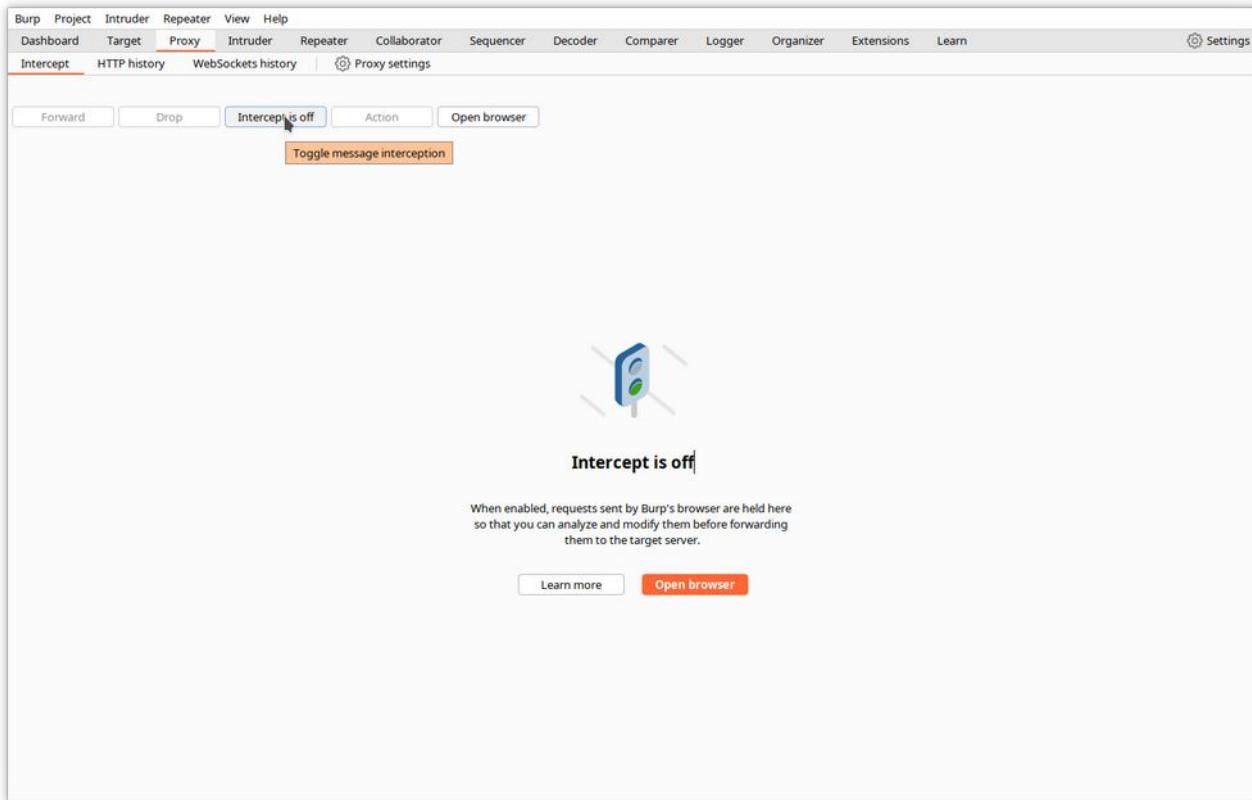
The main interest of Burp is that it can intercept traffic from the browser, and allows you to modify it before delivering it to its target.

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

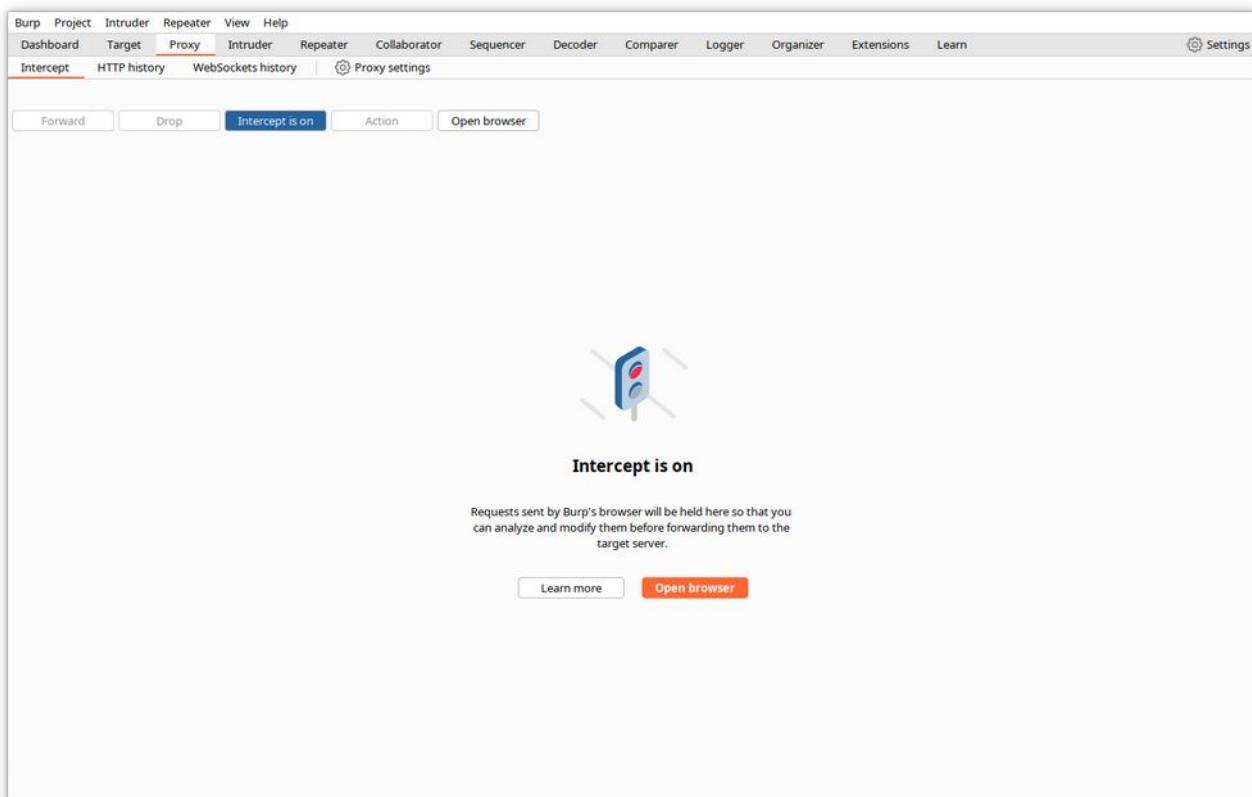


We will now modify the data sent by the POST request.

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.

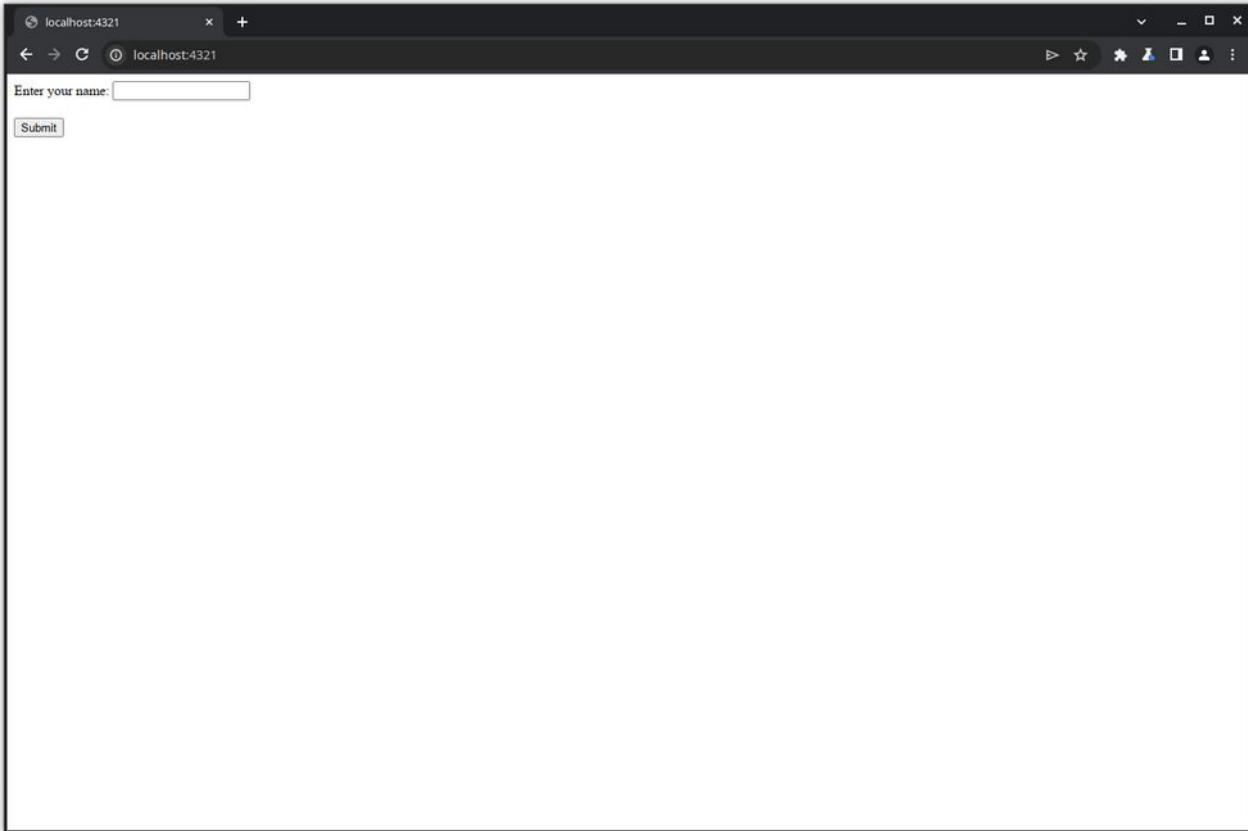
Let's start with a fresh browser. Type your target URL in the address bar and press **<enter>**. The request is sent to Burp and appears in the Intercept tab.

The screenshot shows the PortSwigger homepage. At the top, there is a banner with the text "The latest research into web race conditions". Below the banner, there is a large graphic titled "STATE MACHINE" with various arrows and labels like "login", "good login", "for pas", and "proceed to". Below the graphic, there are three icons: a blue cube, a flask with orange liquid, and a teal hexagon. Below each icon is a link: "Discover the latest research →", "Try the techniques for yourself →", and "Get the latest product capabilities →". At the bottom of the page, there are three sections: "Web Security Academy", "Documentation and support", and "Join the community".

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. In the center, there is a list of requests. The first request is highlighted with the status "Request to http://localhost:4321 [127.0.0.1]". Below the list are buttons for "Forward", "Drop", "Intercept is on" (which is currently selected), "Action", and "Open browser". To the right of the list is the "Inspector" panel, which displays "Request attributes" (2), "Request query parameters" (0), "Request body parameters" (0), "Request cookies" (0), and "Request headers" (15). At the bottom of the interface, there is a search bar and a "0 highlights" indicator.

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.



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

Filter settings: Hiding CSS, image and general binary content

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

**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
```

**Response**

```
1 HTTP/1.0 200 OK
2 Server: BaseHTTP/0.6 Python/3.8.10
3 Date: Wed, 02 Oct 2024 14:24:10 GHT
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"
14         required>
15     <br>
16     <input type="submit" value="Submit">
17   </form>
18 </body>
19 </html>
```

**Inspector**

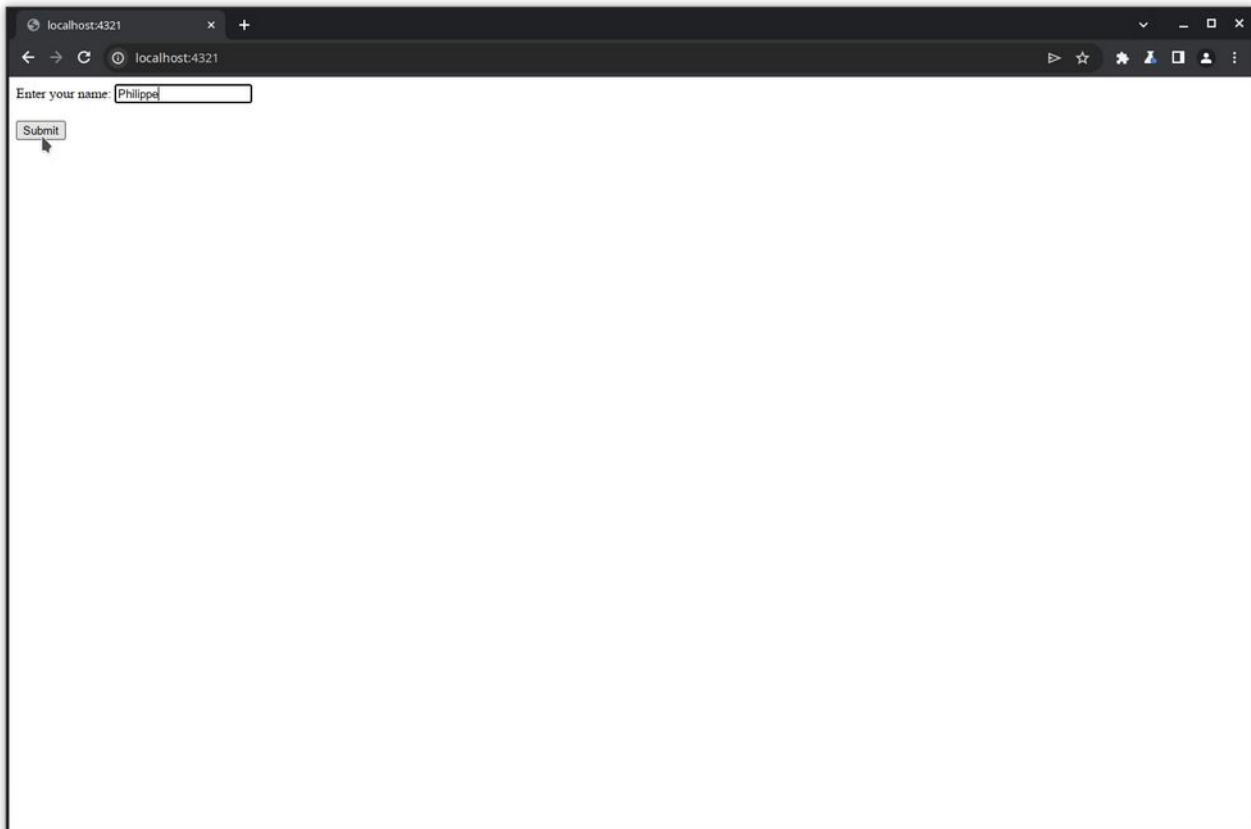
Request attributes 2 ▾

Request headers 15 ▾

Response headers 3 ▾

Notes

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



The request is sent to Burp.

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

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 ?k_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=0.7
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)

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 with the 'Proxy' tab selected. A request to `http://localhost:4321 [127.0.0.1]` is displayed. In the 'Raw' tab, the 'username' header has been modified to 'Gaston'. The 'Inspector' panel on the right shows the selected text 'Gaston' and its decoded form 'Gaston'.

```
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=0.7
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=Gaston
```

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

The screenshot shows a browser window displaying the response from the modified request. The page content is 'Hello, Gaston!'

The screenshot shows the Burp Suite interface. The 'Proxy' tab is active, displaying a list of recent requests. Request 5 is a POST to http://localhost:4321, and Request 4 is a GET to the same endpoint. Both requests return a 200 OK status with HTML content. The 'Response' tab shows the raw HTTP response, which includes the server header, date, content type, and the rendered HTML body: "Hello, Gaston!". The 'Inspector' tab provides detailed information about the response, including request attributes, body parameters, headers, and notes.

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.

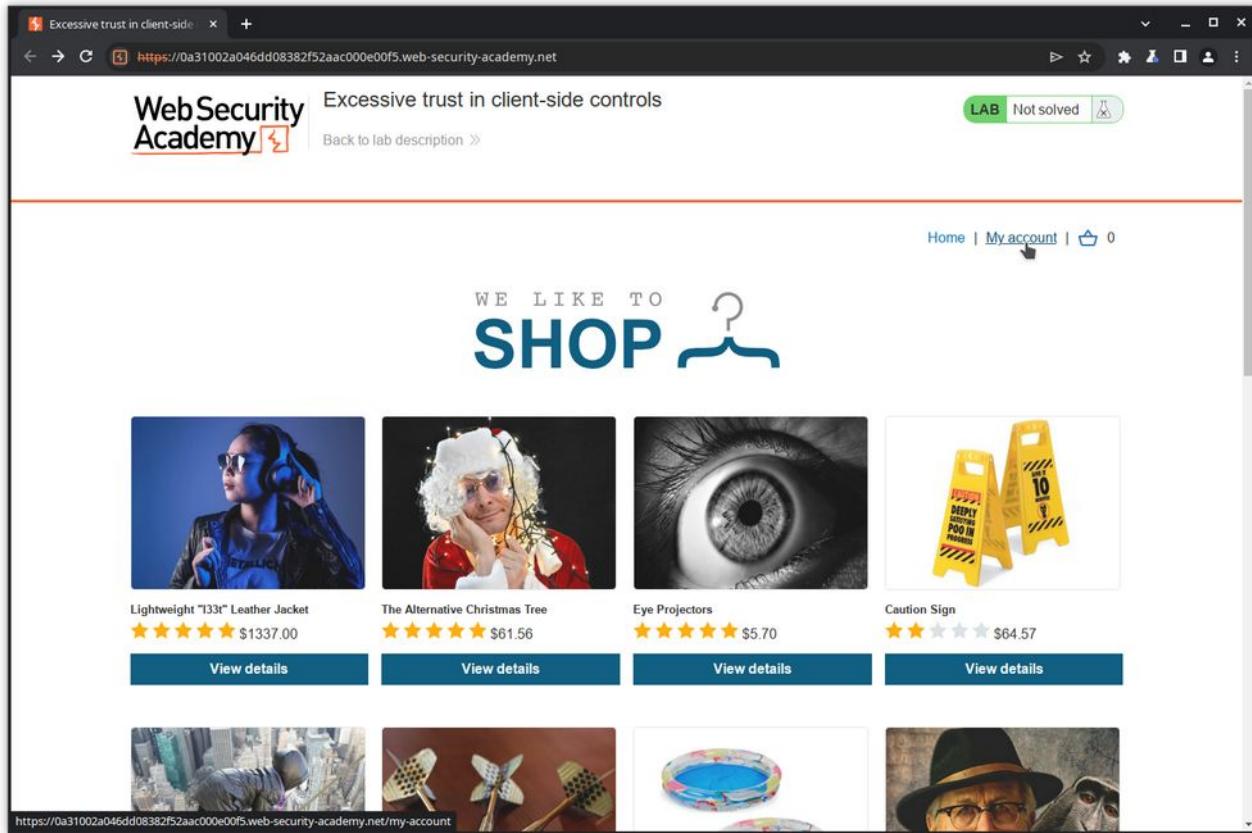
## Using Repeater

### 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.

Let's open the lab page.



Excessive trust in client-side controls

Web Security Academy

Back to lab description >

LAB Not solved

Home | My account | 0

WE LIKE TO  
SHOP

Lightweight "I33t" Leather Jacket  
★★★★★ \$1337.00

The Alternative Christmas Tree  
★★★★★ \$61.56

Eye Projectors  
★★★★★ \$5.70

Caution Sign  
★★★★★ \$64.57

[View details](https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account)

[View details](https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account)

[View details](#)

[View details](#)

## Step 2: Log in

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

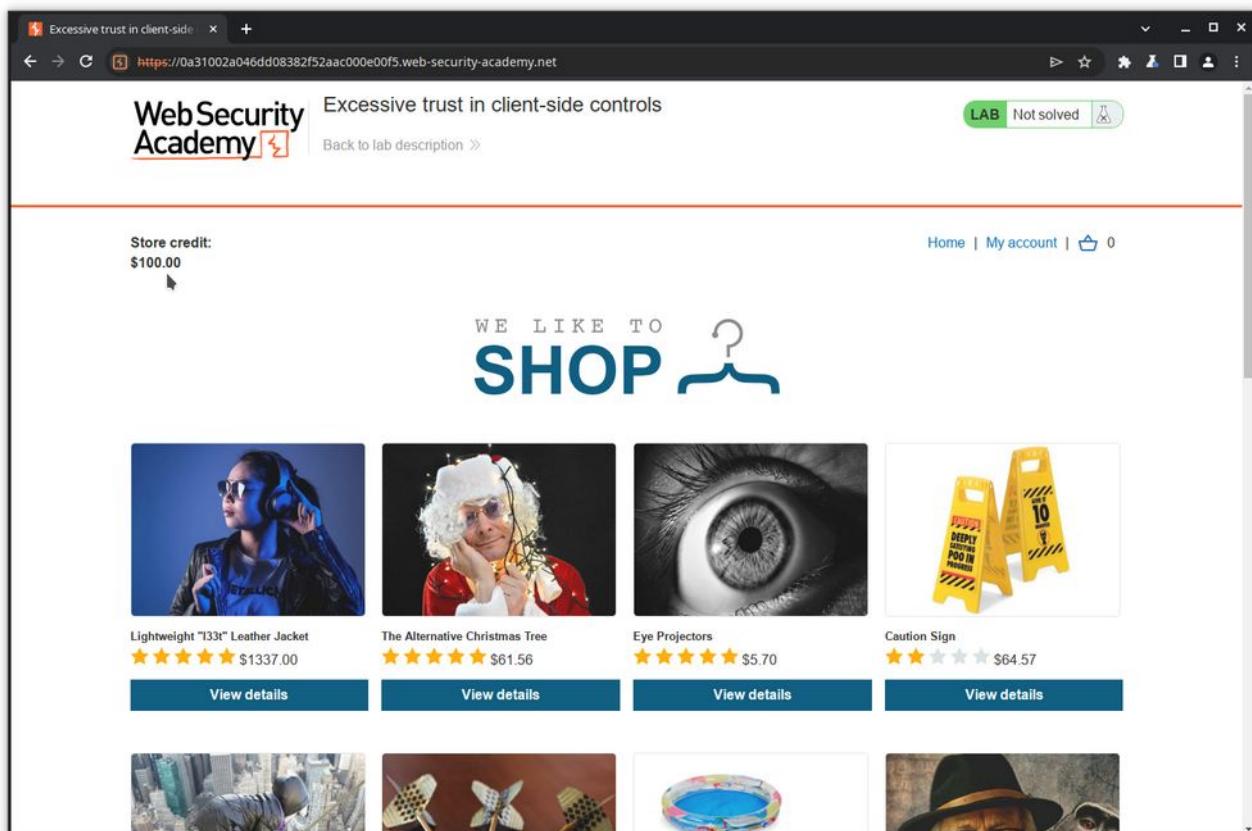
The screenshot shows a web browser window for the 'Excessive trust in client-side controls' lab on the Web Security Academy. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/login>. The page title is 'Excessive trust in client-side controls'. At the top right, there is a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

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

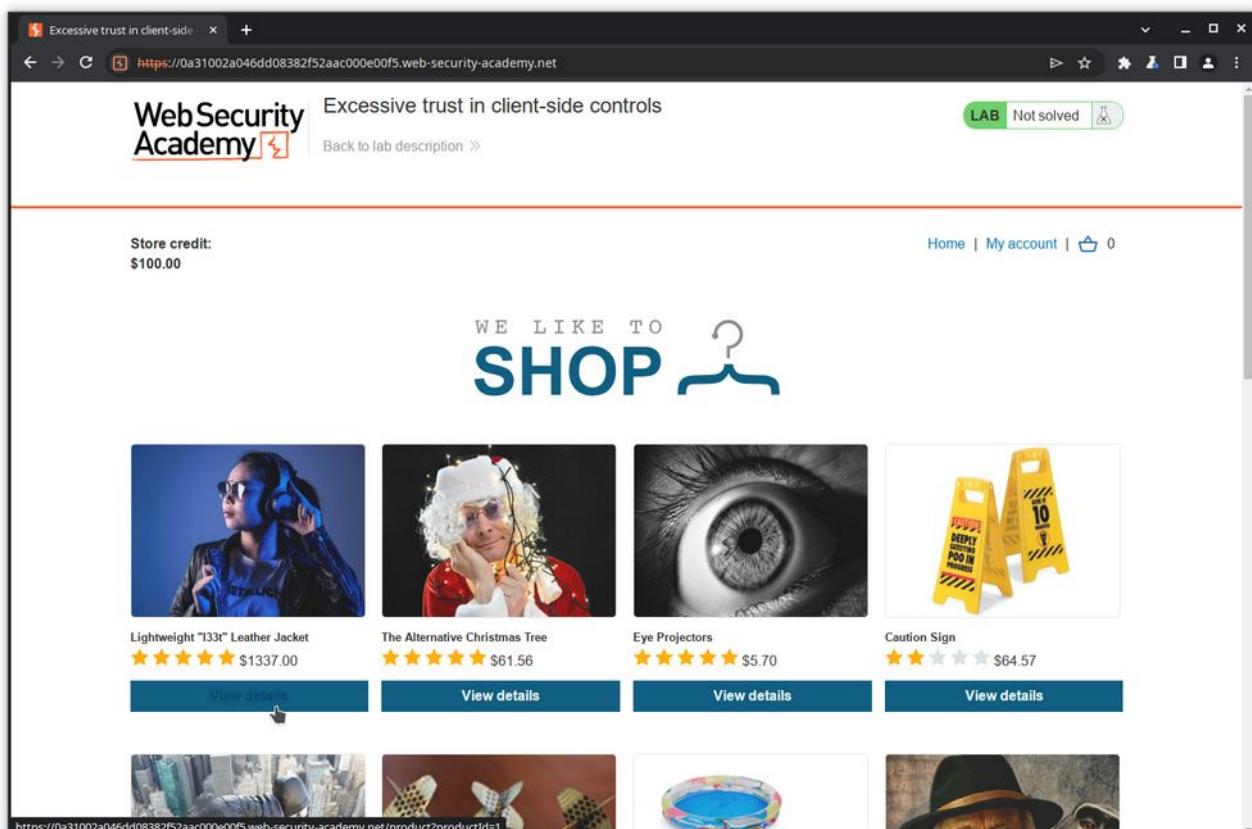
The screenshot shows a web browser window for the 'My Account' page of the 'Excessive trust in client-side controls' lab. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account?id=wiener>. The page title is 'Excessive trust in client-side controls'. At the top right, there is a green 'LAB' button, a 'Not solved' status, and a test tube icon. Below the title, there is a link 'Back to lab description >'. On the left, it displays 'Store credit: \$100.00'. At the top right, there are links for 'Home', 'My account', a shopping cart icon showing '0' items, and 'Log out'. In the center, there is a section titled 'My Account' with the message 'Your username is: wiener'. Below this, there is a form with an 'Email' field and a green 'Update email' button.

### Step 3: Try to purchase an item

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



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

Store credit:  
\$100.00

Lightweight "I33t" Leather Jacket

★★★★★  
\$1337.00

Home | My account | 1

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 a single item: a 'Lightweight "I33t" Leather Jacket' priced at \$1337.00. The quantity is set to 1. Below the cart, there is a coupon input field with an 'Apply' button. The total amount shown is \$1337.00. At the bottom, a 'Place order' button is visible. The status bar at the top right indicates 'LAB Not solved'.

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

The screenshot shows the same browser window after attempting to place the order. A red error message 'Not enough store credit for this purchase' is displayed above the cart. The rest of the interface is identical to the successful attempt, showing the cart contents, coupon section, total price, and 'Place order' button.

## Step 4: Examine the purchase actions

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

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

- Proxy Tab:** Selected, showing a list of captured requests and responses.
- Selected Request:** A POST request to `/cart?productID=1`.
- Selected Response:** An HTML page containing a form for adding a product to the cart. The form includes hidden fields for `productId` and `price`.
- Inspector Panel:** Shows the selected text from the response, which includes the following code snippet:
 

```
<input required type="hidden" name="productId" value="1">
<input required type="hidden" name="price" value="133700" type="hidden">
```
- Request and Response Headers:** Detailed in the Inspector panel.

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

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-side controls	✓	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-side controls	✓	79.125.84.16	
140	https://0a31002a046dd08382f5...	GET	/product?productId=1		✓	200	5283	HTML		Excessive trust in client-side controls	✓	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-side controls	✓	79.125.84.16	
137	https://0a31002a046dd08382f5...	GET	/			200	11075	HTML		Excessive trust in client-side controls	✓	79.125.84.16	
136	https://0a31002a046dd08382f5...	GET	/my-account?id=wiener		✓	200	3674	HTML		Excessive trust in client-side controls	✓	79.125.84.16	
135	https://0a31002a046dd08382f5...	POST	/login		✓	302	188				✓	79.125.84.16	

**Request**

Pretty Raw Hex

```
1 POST /cart HTTP/2
2 Host: 0a31002a046dd08382f5aac000e00f5.web-security-academy.net
3 Cookie: session=INC5PTzKugdVttXx0AxI4u3RBEKyqrZR
4 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
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
10 Origin: https://0a31002a046dd08382f5aac000e00f5.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: 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-Dest: iframe
17 Sec-Fetch-Dest: document
18 Referer: https://0a31002a046dd08382f5aac000e00f5.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, l=2
22 productId=1&redirect=PRODUCT&quantity=1&price=133700
```

**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
```

**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

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

https://0a31002a046dd08382f5aac000e00f5.web-security-academy.net/cart?err=INSUFFICIENT\_FUNDS

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

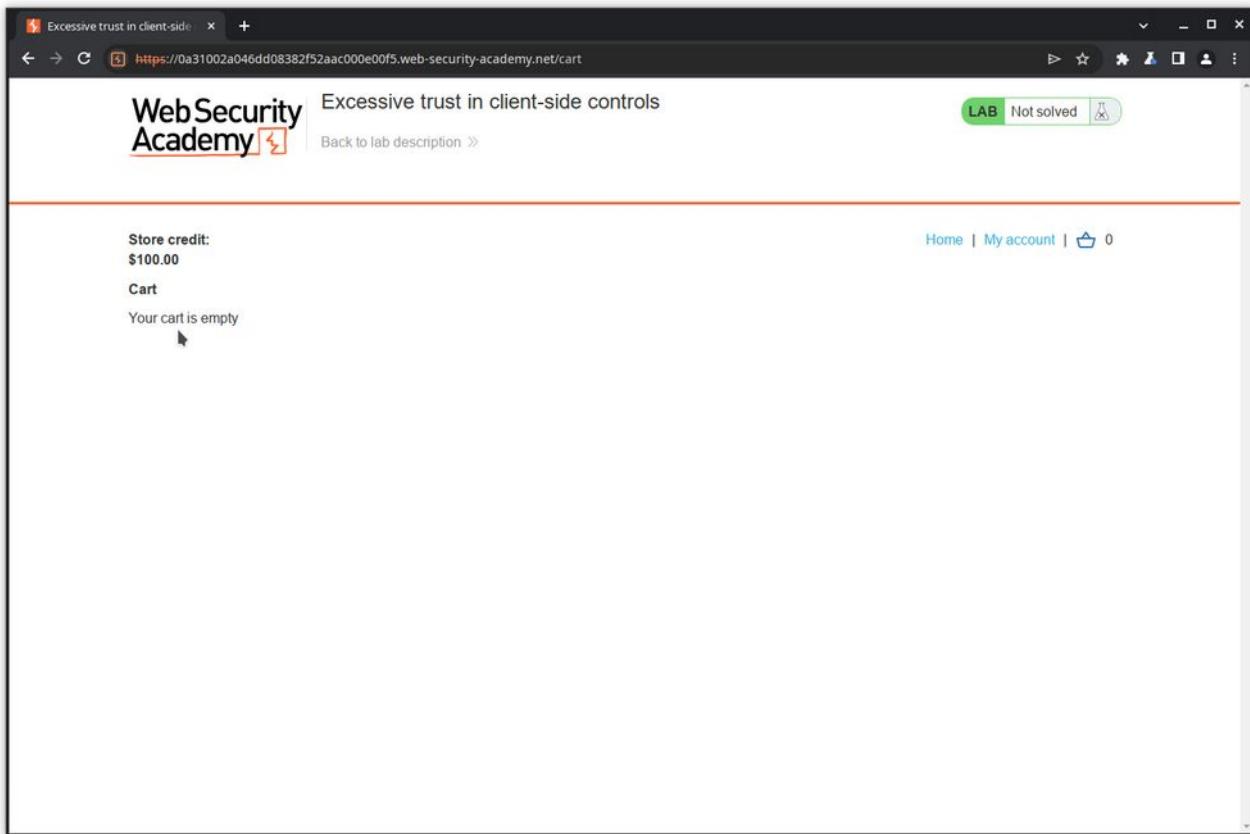
Name	Price	Quantity
Lightweight "I33I" Leather Jacket	\$1337.00	- 1 + Remove

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**.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The left pane displays a list of network requests, and the right pane shows the 'Inspector' tool. A context menu is open over the last request in the list, with the option 'Send to Repeater' highlighted.

#	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			5283	HTML			Excessive trust in client...	✓	79.125.84.16	
137	http://1a31002a046dd08382f5...	GET	/			11075	HTML			Excessive trust in client...	✓	79.125.84.16	

**Request**

```

Pretty Raw Hex
1 POST /cart HTTP/2
2 Host: 0a31002a046dd08382f52aac000e00f5.
3 Cookie: session=INC5PTskugdVttXx0Ax14u3RBKyqcrZ
4 Content-Length: 49
5 Content-Type: application/x-www-form-urlencoded
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.
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/119.0.0.145.159 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;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-Dest: iframe
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a31002a046dd08382f52aac000e00f5./product?productId=1
20 Accept-Encoding: gzip, deflate, br
21 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
22 Priority: u=0, i
23 productId=1&redirect=PRODUCT&quantity=1&price=133700

```

**Inspector**

```

Selected text
productId=1&redirect=PRODUCT&quantity=1
&price=133700

```

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

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The left pane displays the copied message from the previous screenshot. The right pane shows the 'Inspector' tool with the selected text 'productId=1&redirect=PRODUCT&quantity=1&price=133700'.

**Request**

```

Pretty Raw Hex
1 POST /cart HTTP/2
2 Host: 0a31002a046dd08382f52aac000e00f5.web-security-academy.net
3 Cookie: session=INC5PTskugdVttXx0Ax14u3RBKyqcrZ
4 Content-Length: 49
5 Content-Type: application/x-www-form-urlencoded
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/119.0.0.145.159 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;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-Dest: iframe
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/product?productId=1
20 Accept-Encoding: gzip, deflate, br
21 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
22 Priority: u=0, i
23 productId=1&redirect=PRODUCT&quantity=1&price=133700

```

**Inspector**

```

Selected text
productId=1&redirect=PRODUCT&quantity=1
&price=133700

```

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

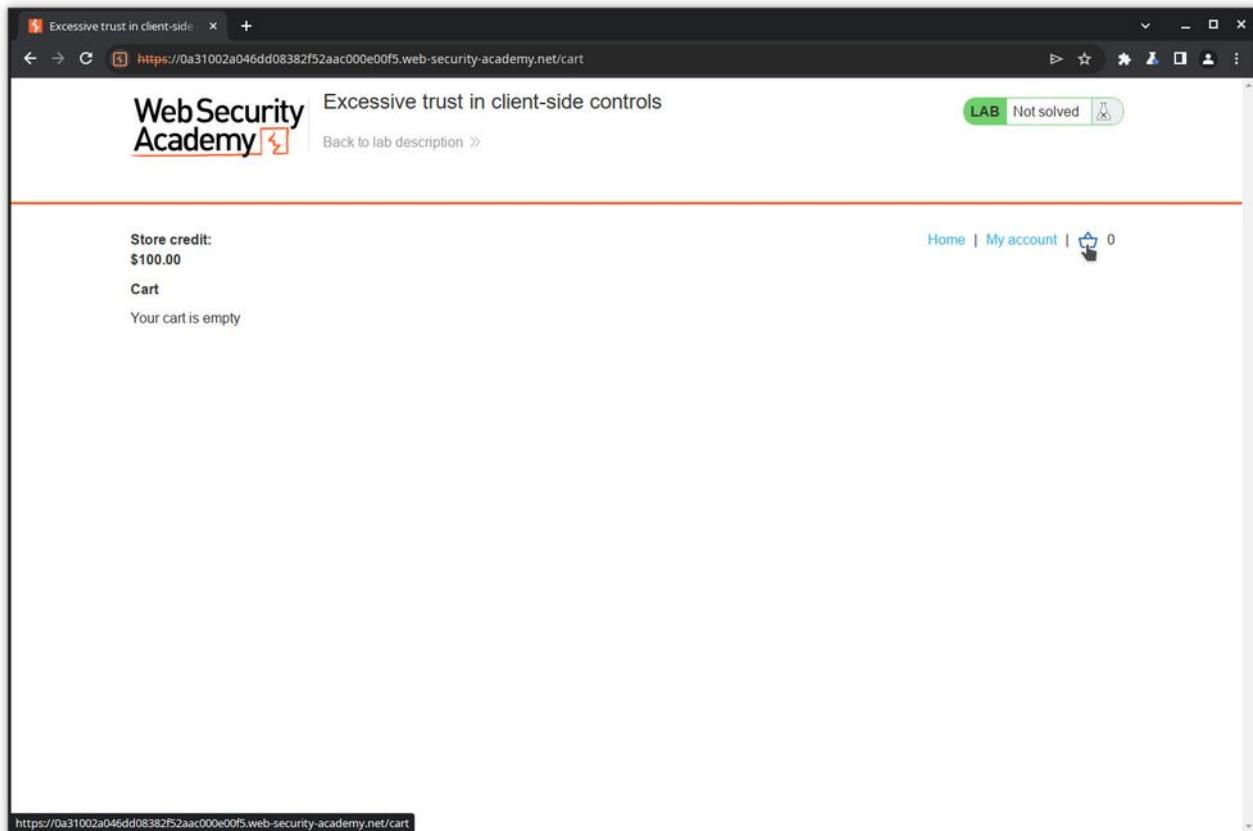
The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a POST request is shown with the URL https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net. The body of the request contains the parameter productId=1&redirect=PRODUCT&quantity=1&price=1. In the Response pane, the server's response is displayed, showing a 302 Found status code and a Location header pointing back to the same URL. The Inspector pane on the right highlights the selected text productId=1&redirect=PRODUCT&quantity=1&price=1.

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

This screenshot is identical to the one above, showing the Burp Suite Repeater tab. A POST request is made to https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net with the parameter productId=1&redirect=PRODUCT&quantity=1&price=1. The response shows a 302 Found status and a Location header. The Inspector pane highlights the selected text productId=1&redirect=PRODUCT&quantity=1&price=1.

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

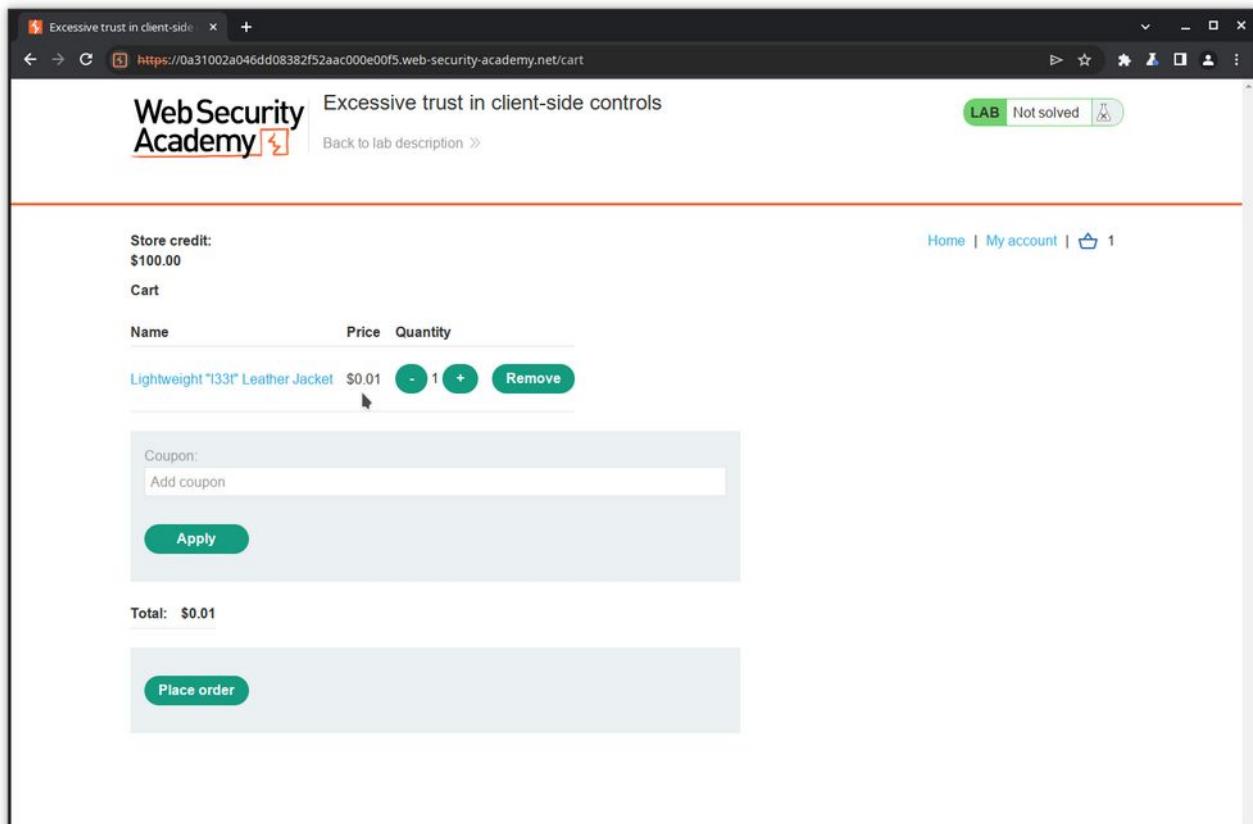
Back to lab description >

Store credit:  
\$100.00

Cart  
Your cart is empty

Home | My account |  0

Let's check our cart.



Excessive trust in client-side controls

Back to lab description >

Store credit:  
\$100.00

Cart

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

Coupon:  
  


Total: \$0.01



**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.

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

Congratulations, you solved the lab!

Share your skills! Continue learning >

Store credit:  
\$99.99

Your order is on its way!

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

Total: \$0.01

LAB 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>.

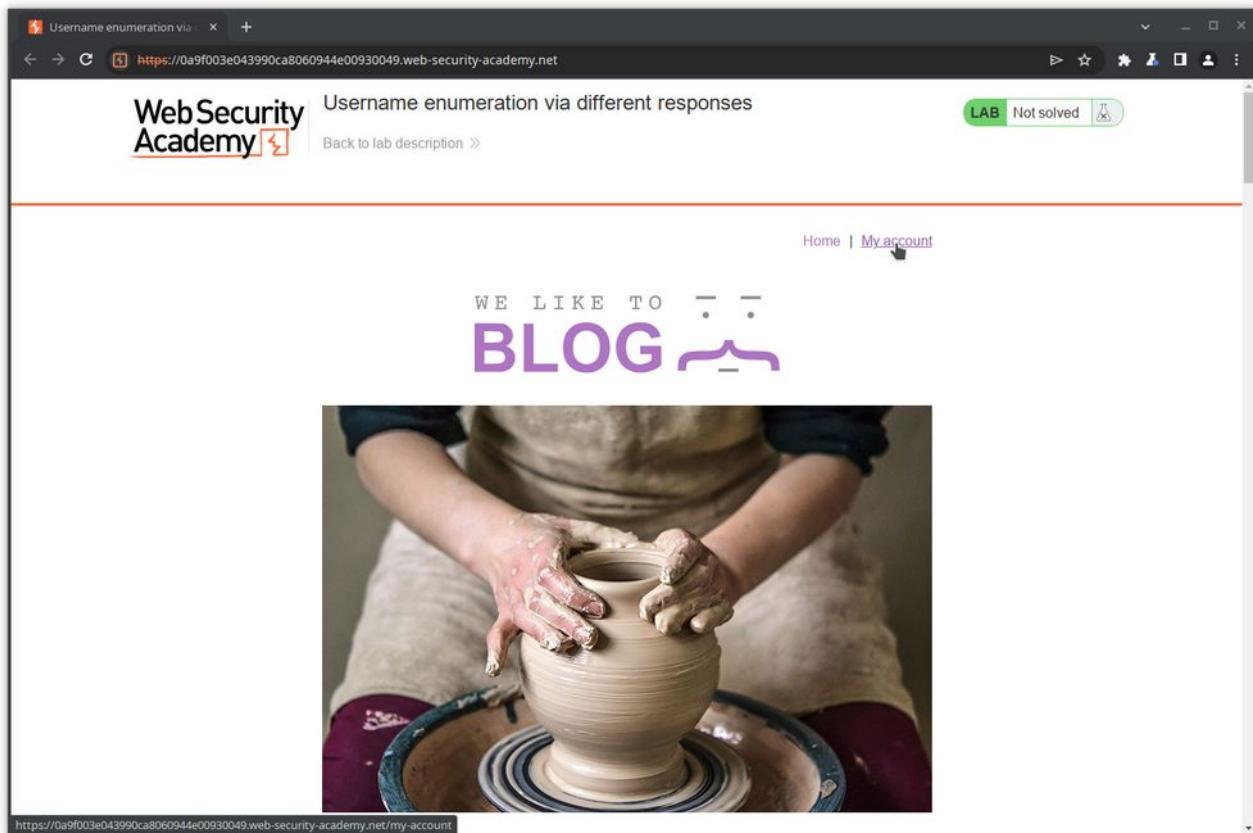
Burp Intruder is a tool that allows sending the same HTTP request several times, inserting different payloads into predefined positions each time. In this example, we will use it to brute force the username and password to get access to a web application.

### 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**.

Username enumeration via different responses

Back to lab description >

Home | My account

## Login

Username  
Philippe

Password  
...

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*.

Username enumeration via different responses

Back to lab description >

Home | My account

## Login

Invalid username

Username

Password

Log in

## 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 with the 'Proxy' tab selected. The 'HTTP history' section lists several requests, including a POST to '/login'. A context menu is open over this POST request, with the 'Send to Intruder' option highlighted. To the right of the proxy history, there's an 'Inspector' panel showing details like 'Selected text: Philippe' and 'Request headers'.

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 request is displayed with several payload positions highlighted in red. The 'Payload positions' section below the request indicates there is 1 payload position. The 'Start attack' button is visible at the top right.

```

1 POST /login HTTP/2
2 Host: 0a9f003e043990ca8060944e00930049.web-security-academy.net
3 Cookie: session=VtfPOTK5VEcFahOabIdXH4Y0yinvfQLB
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 10
6 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"
7 Sec-Ch-Ua: "Chromium";v="119", "Not ?A_Brand";v="24"
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: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
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: u0,i
22
23 username=$Philippe$&password=abc
    
```

1 payload position

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 'Payloads' tab selected. It displays payload sets, settings for a simple list, and processing rules. The 'Load items from file' option is highlighted in the payload settings section.

**Payload sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

**Payload settings [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

**Payload processing**

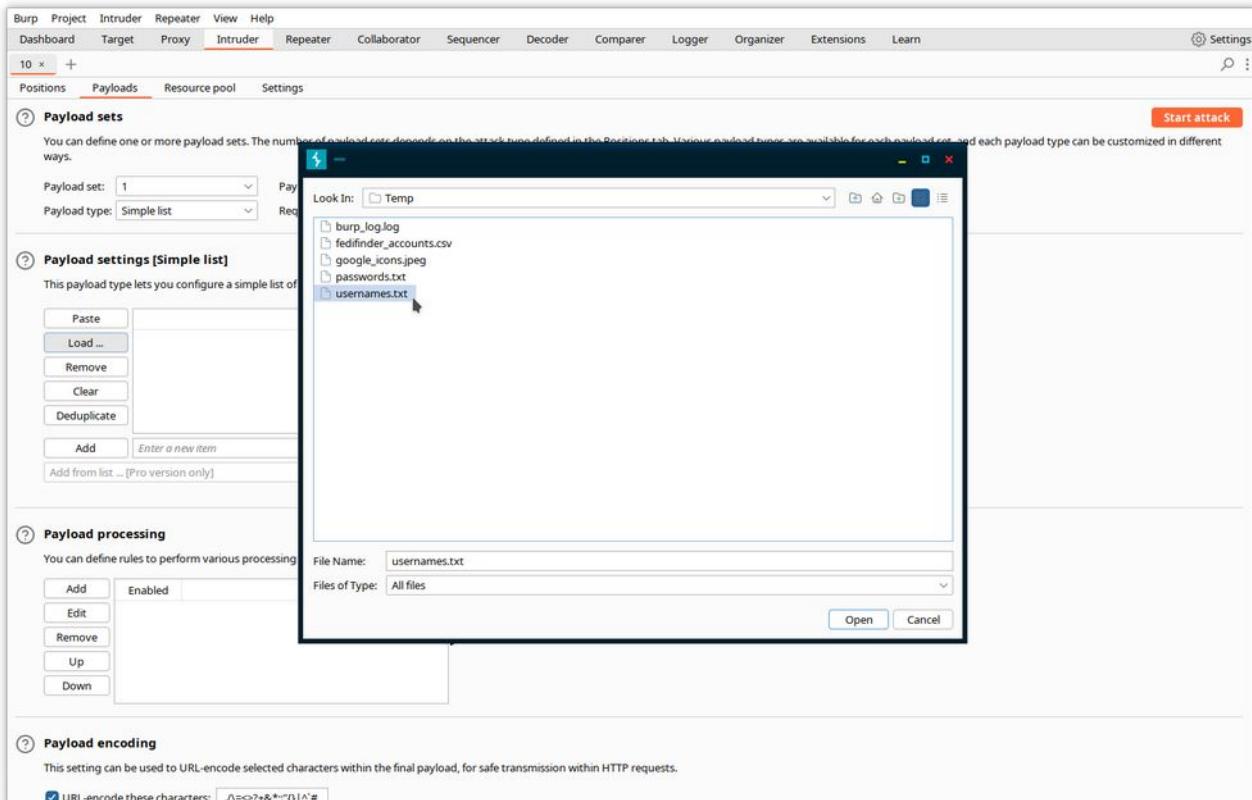
You can define rules to perform various processing tasks on each payload before it is used.

**Payload encoding**

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

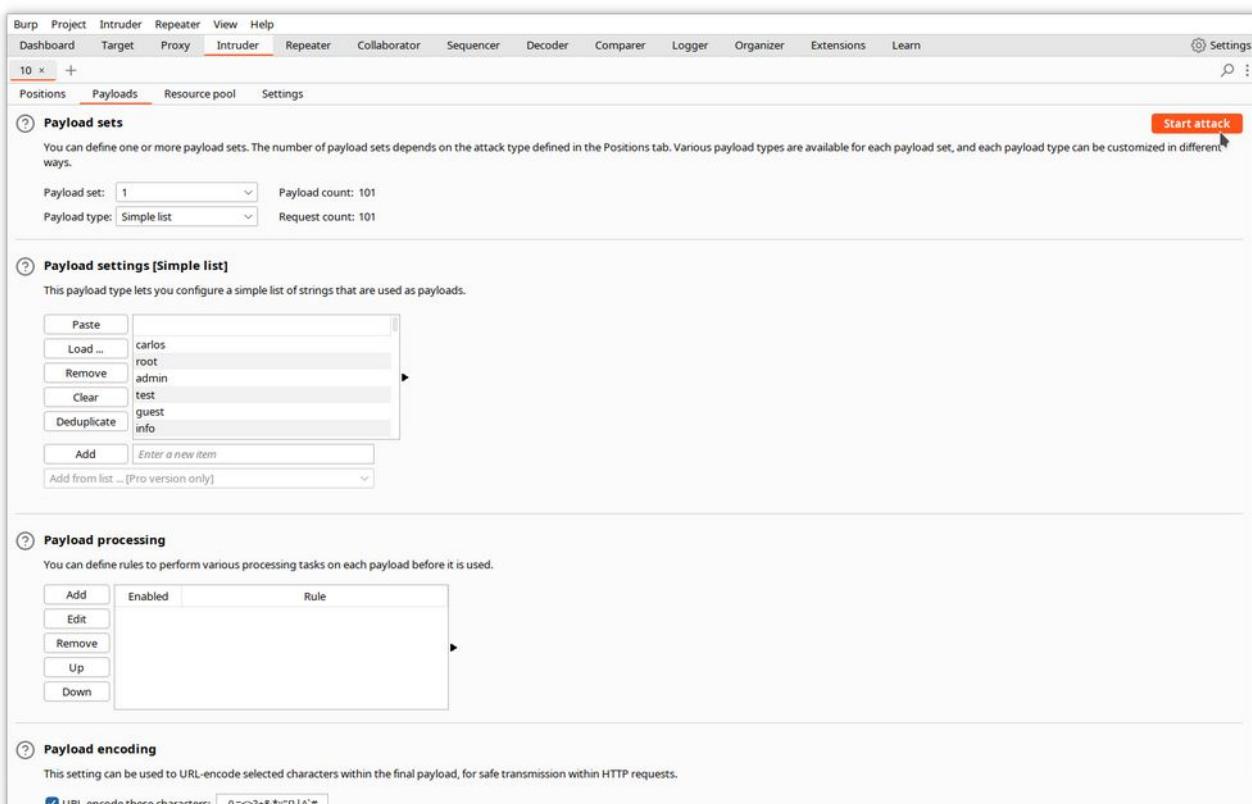
URL-encode these characters: /<>?\*&^{}|^#

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.



The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload sets' section, a 'Simple list' payload type is chosen. A file selection dialog is overlaid on the interface, showing a list of files in the 'Temp' directory. The file 'usernames.txt' is highlighted with a mouse cursor. The dialog includes fields for 'File Name:' set to 'usernames.txt' and 'Files of Type:' set to 'All files'. Buttons for 'Open' and 'Cancel' are at the bottom right.

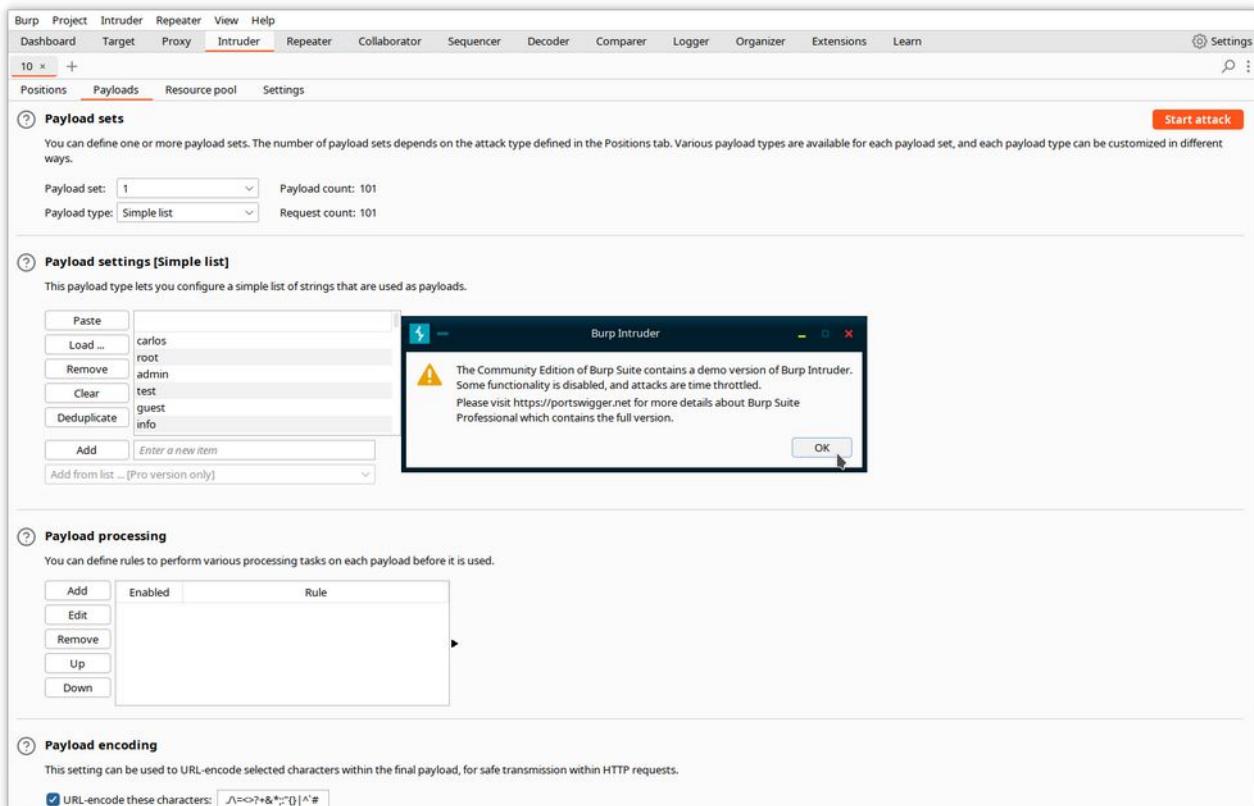
Note that the **Payload count** and **Request count** values are updated with the number of entries in the list.



The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the 'Payload sets' section, a 'Simple list' payload type is chosen. The 'Payload count' and 'Request count' fields show the value '101'. The 'Payload settings [Simple list]' section displays a list of payloads: 'carlos', 'root', 'admin', 'test', 'guest', and 'info'. Below this, the 'Payload processing' section shows a table with one row, and the 'Payload encoding' section shows a URL encoding configuration.

## 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 main pane, there's a section for 'Payload sets' where a payload set of 1 and a payload type of 'Simple list' are configured. Below this, 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 payload type lets you configure a simple list of strings used as payloads. To the right, 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 at the bottom of the modal. Other sections visible include 'Payload processing' and 'Payload encoding'.

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.

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	

Request Response

Pretty Raw Hex

```
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:
    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://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 Connection: keep-alive
23
24 username=adserver&password=abc
```

?

Search

0 highlights

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 during an attack. 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 'adserver' in the Payload column and '200' in the Status code column. Below this table is another section titled 'Request Response'. It has tabs for 'Pretty' (which is selected), 'Raw', 'Hex', and 'Render'. The 'Pretty' tab shows a snippet of HTML code with line numbers 47 through 57. Line 53 contains the text 'Incorrect password'. Below the code editor are icons for help, settings, back, forward, and search, along with a status bar that says 'Finished' and '0 highlights'.

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. In the main pane, there's a 'Payload sets' section where a simple list of credentials has been loaded. Below it is a 'Payload settings [Simple list]' section. Further down is a 'Payload processing' section with a table for rules. On the right, a modal dialog asks 'Do you want to save this attack?' with options to 'Keep in memory' or 'Discard'. The bottom status bar shows '0 highlights'.

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' tab is active, displaying a list of captured requests. One request, a POST to '/login', has its password field highlighted in red. The 'Inspector' panel on the right shows the selected text 'abc' and various context menu options for the highlighted field.

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 type' dropdown is set to 'Simple list'. A file selection dialog is open, showing files like 'burp\_log.log', 'feedfinder\_accounts.csv', 'google\_icons.jpeg', 'passwords.txt', and 'usernames.txt'. The 'passwords.txt' file is selected, and the 'Open selected file' button is highlighted with a red box.

Then, start the new attack.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payload sets' section now shows a payload count of 100 and a request count of 100. The 'Payload settings [Simple list]' section contains 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 has a checked checkbox for URL-encoding specific characters.

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.

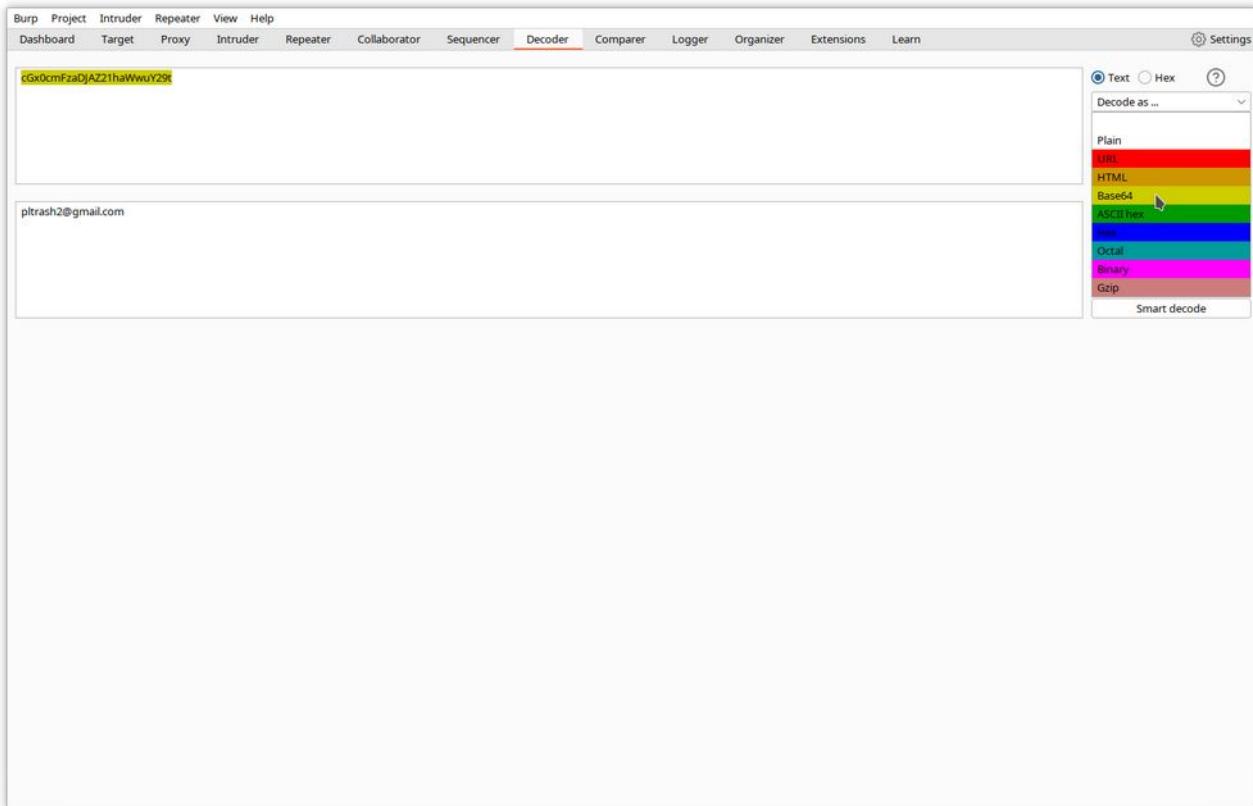
Let's confirm in the browser by performing an interactive login with the found credentials.

Success!!

## 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			3250	
0		200			3248	
2	root	200			3248	
1	carlos	200			3248	
3	admin	200			3248	
5	guest	200			3248	
4	test	200			3248	
6	info	200			3248	
8	mysql	200			3248	
7	adm	200			3248	
9	user	200			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: 3142
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">
```

?

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    </head>
15    <body>
16      <script src="/resources/labheader/js/labHeader.js">
17      </script>
18      <div id="academyLabHeader">
```

?

Search

Finished

Scan

- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Computer**
- 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

0 highlights

Message editor documentation

Intruder results documentation

The screenshot shows two captured messages in the Burp Suite interface. Both messages have a length of 3,250 bytes.

**Message 1 (Left):**

```
<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>
<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="/>Home</a><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>
```

**Comparison Table:**

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

**Buttons and Options:**

- Key: Modified Deleted Added
- Sync views
- Compare ... Words
- Perform a word-level compar