

# 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.

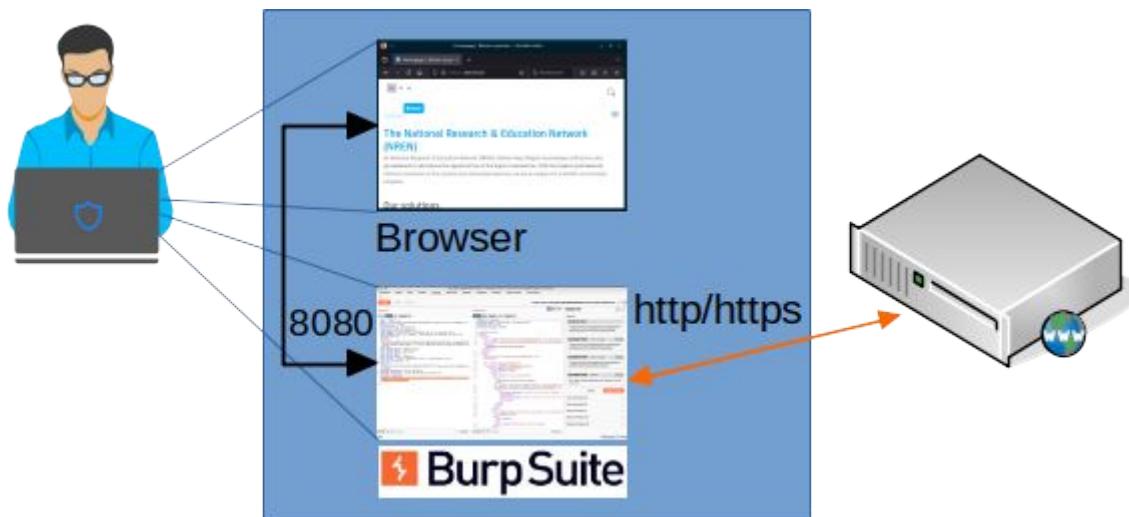
Burp Suite is currently available in three editions:

- 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.

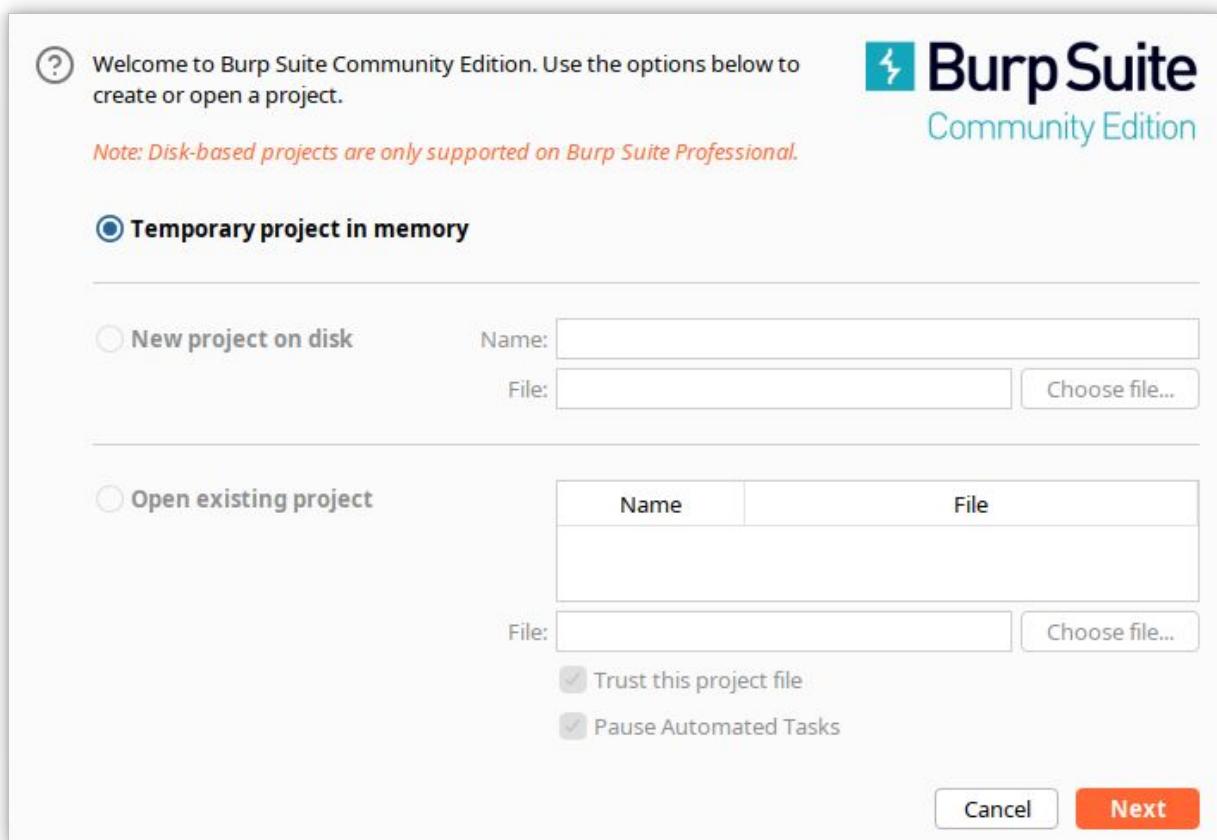
## Monitoring web traffic

The following description is based on the free Community Edition.

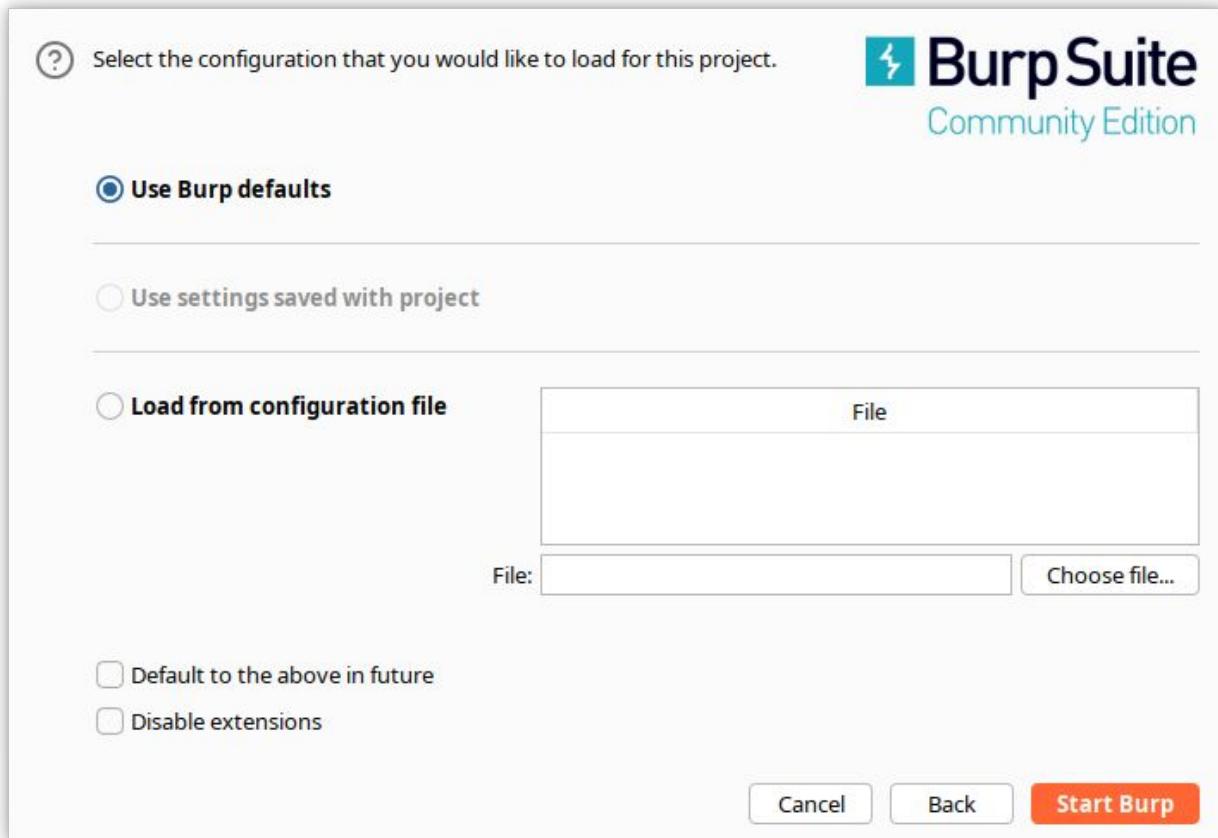
### Starting Burp Suite

From the menu, start Burp Suite.

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



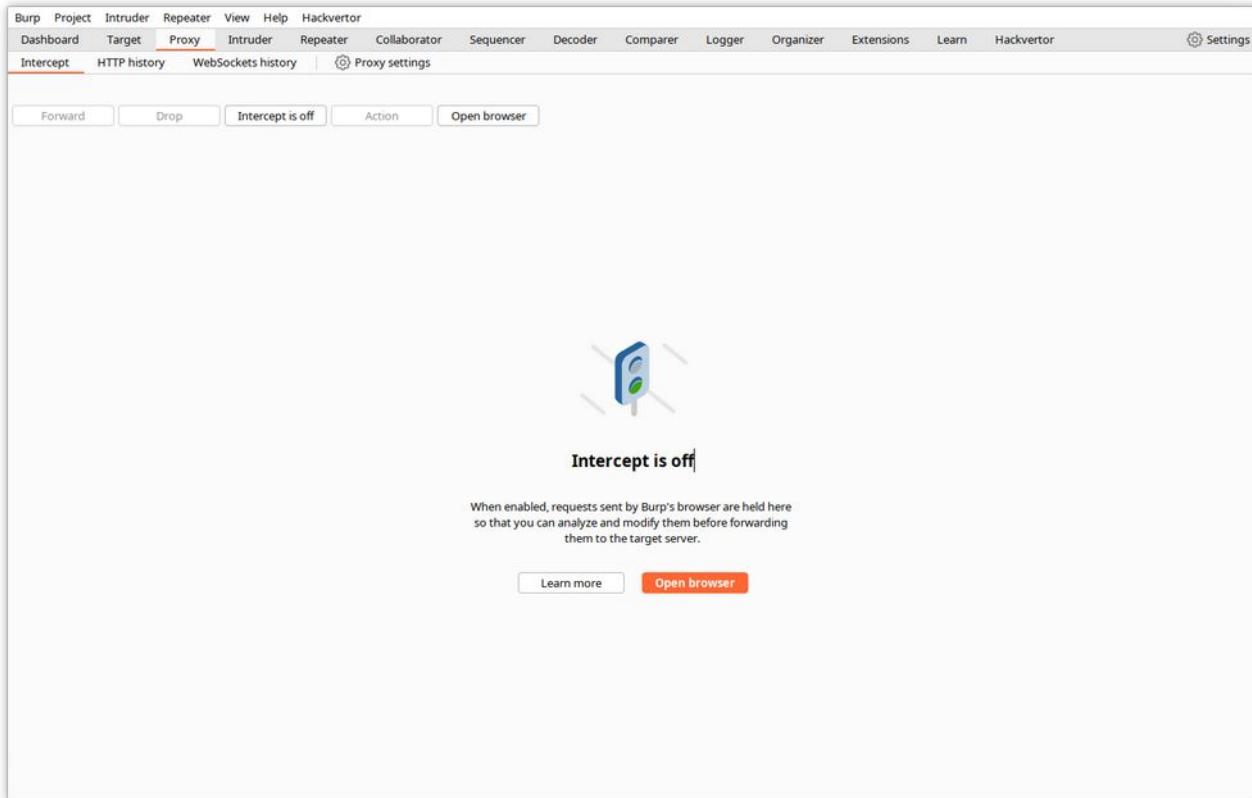
Click **Next**. Burp now asks for a configuration file.



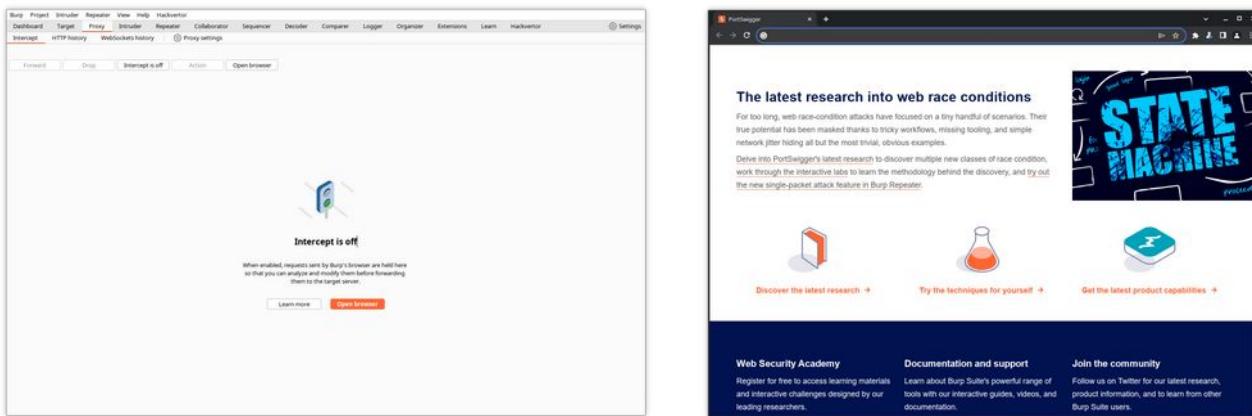
Click **Start Burp**. The *Dashboard* screen opens.

The screenshot shows the Burp Suite Dashboard. The top navigation bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', 'Help', 'Hackvator', 'Dashboard' (selected), 'Target', 'Proxy', 'Intruder', 'Repeater', 'Collaborator', 'Sequencer', 'Decoder', 'Comparer', 'Logger', 'Organizer', 'Extensions', 'Learn', and 'Hackvator'. A 'Settings' icon is also present. The main interface consists of several panels: 'Issue activity [Pro version only]' (listing various security issues like Suspicious input transformation, SMTP header injection, etc.), 'Event log' (listing events such as 'Proxy service started on 127.0.0.1:8080'), and a central 'Advisory' panel which is currently empty. At the bottom, system status is shown: Memory: 258.4MB and Disk: 128.5MB.

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.



## 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' panel 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-GB,en-US;q=0.9,en;q=0.8
16 Connection: close
17
18
```

The 'Response' panel 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 GRT
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 'Inspector' panel on the right shows:

- Request attributes: 2
- Request headers: 15
- Response headers: 3

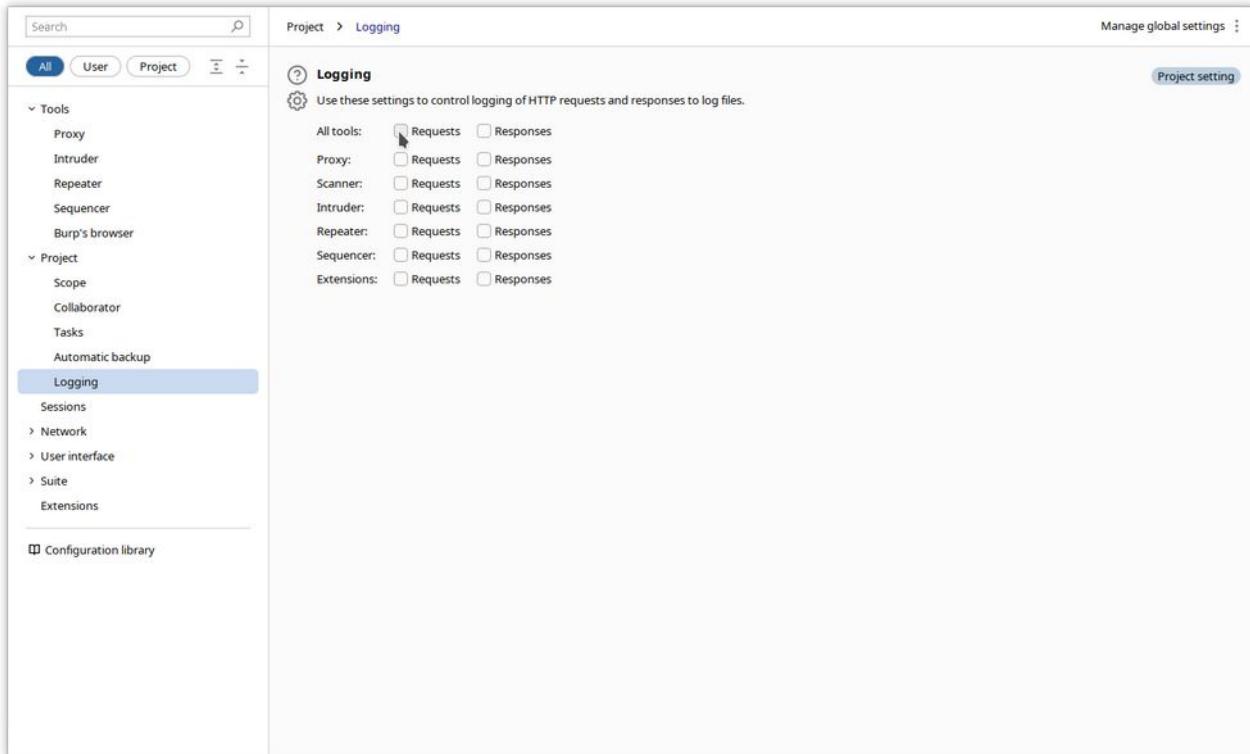
The captured data for the request contain the HTTP request type (GET), the requested path (/) and the complete headers (Host, Cache-Control, User-Agent, accepted document types...), the cookies and the variable values (for POST requests).

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

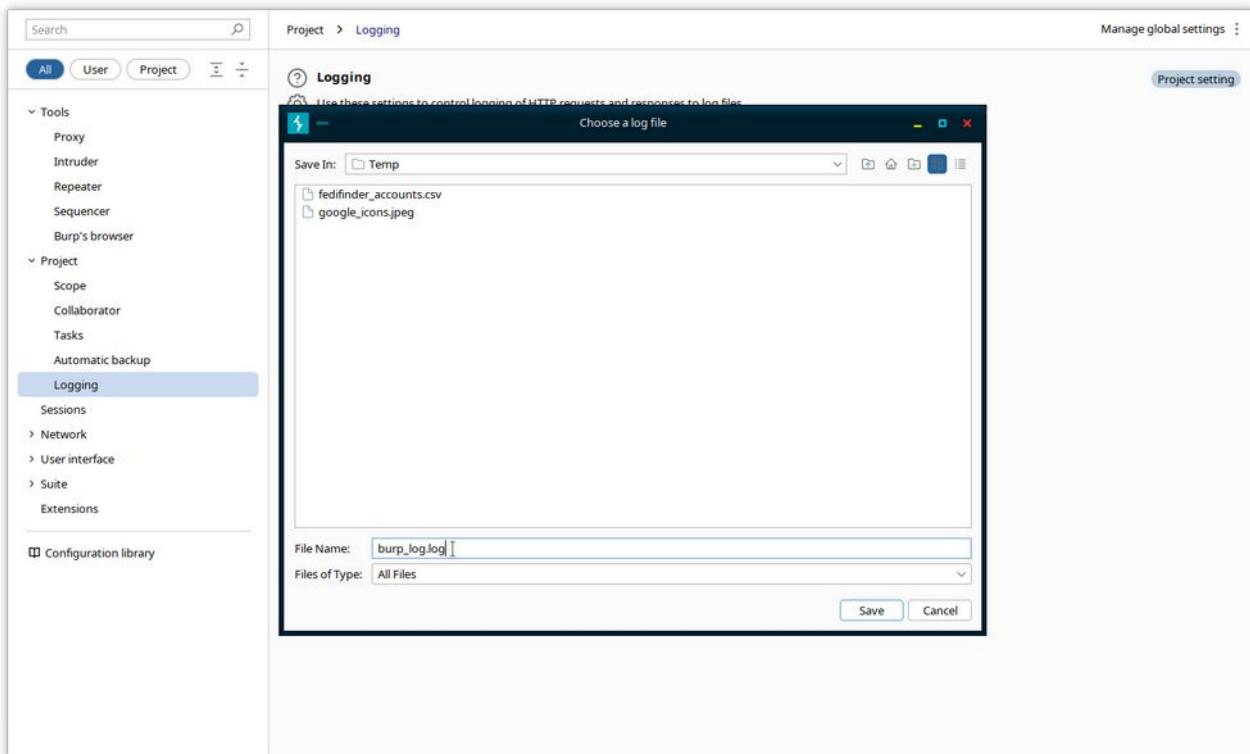
## Logging traffic

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

=====

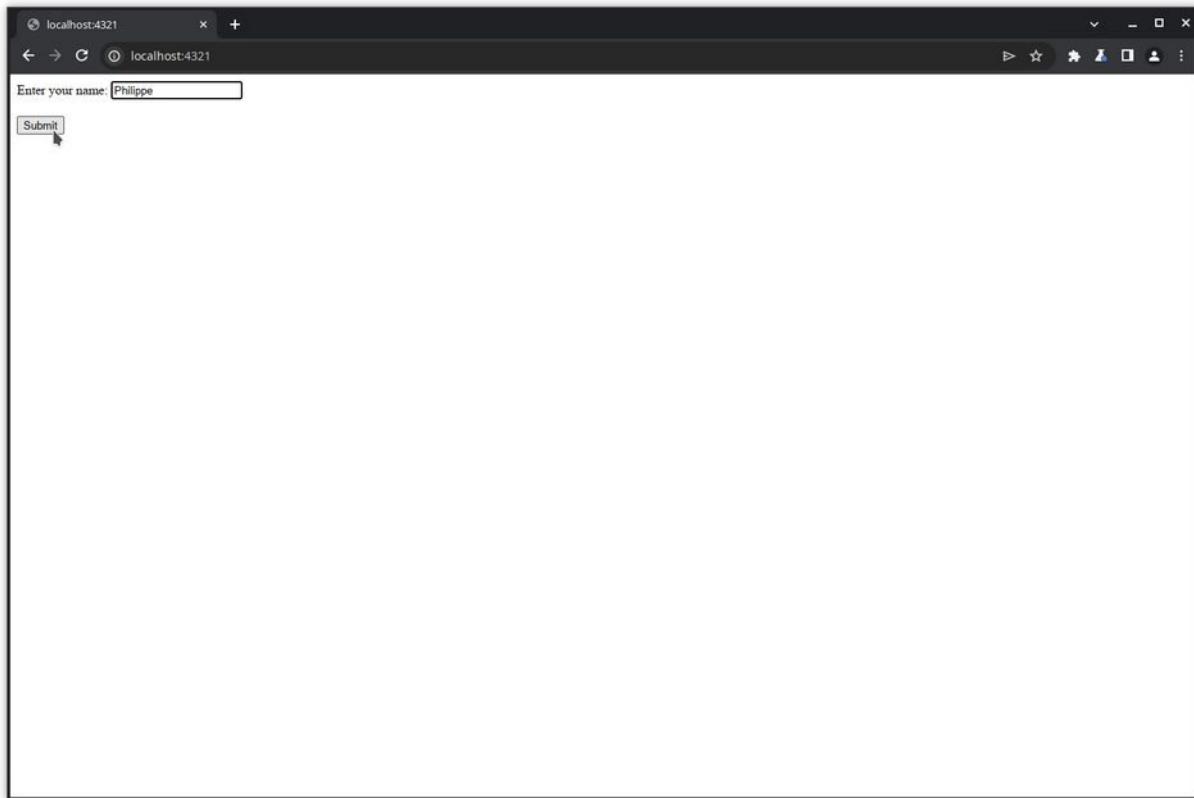
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.8.10
Date: Tue, 01 Oct 2024 17:03:15 GMT
Content-Type: text/html

<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

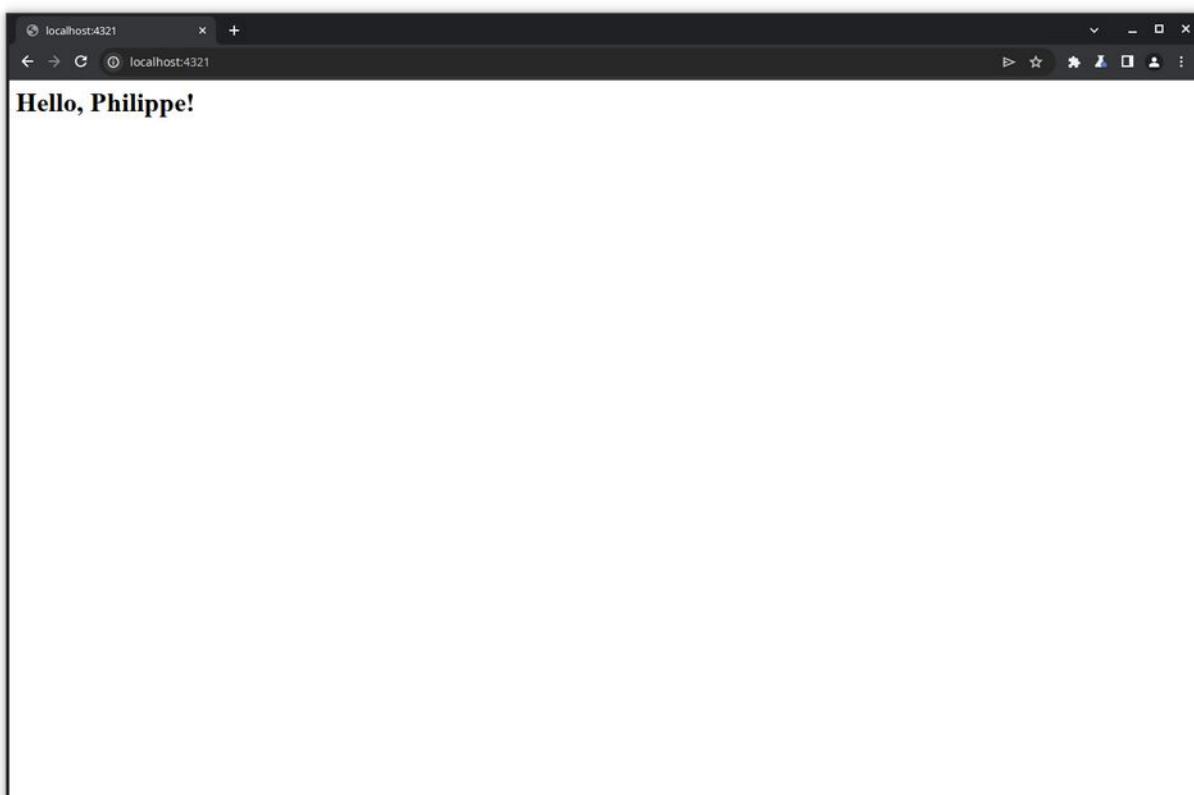
The main interest of Burp is that it can intercept traffic from the browser, and allows you to modify it before delivering it to its target.

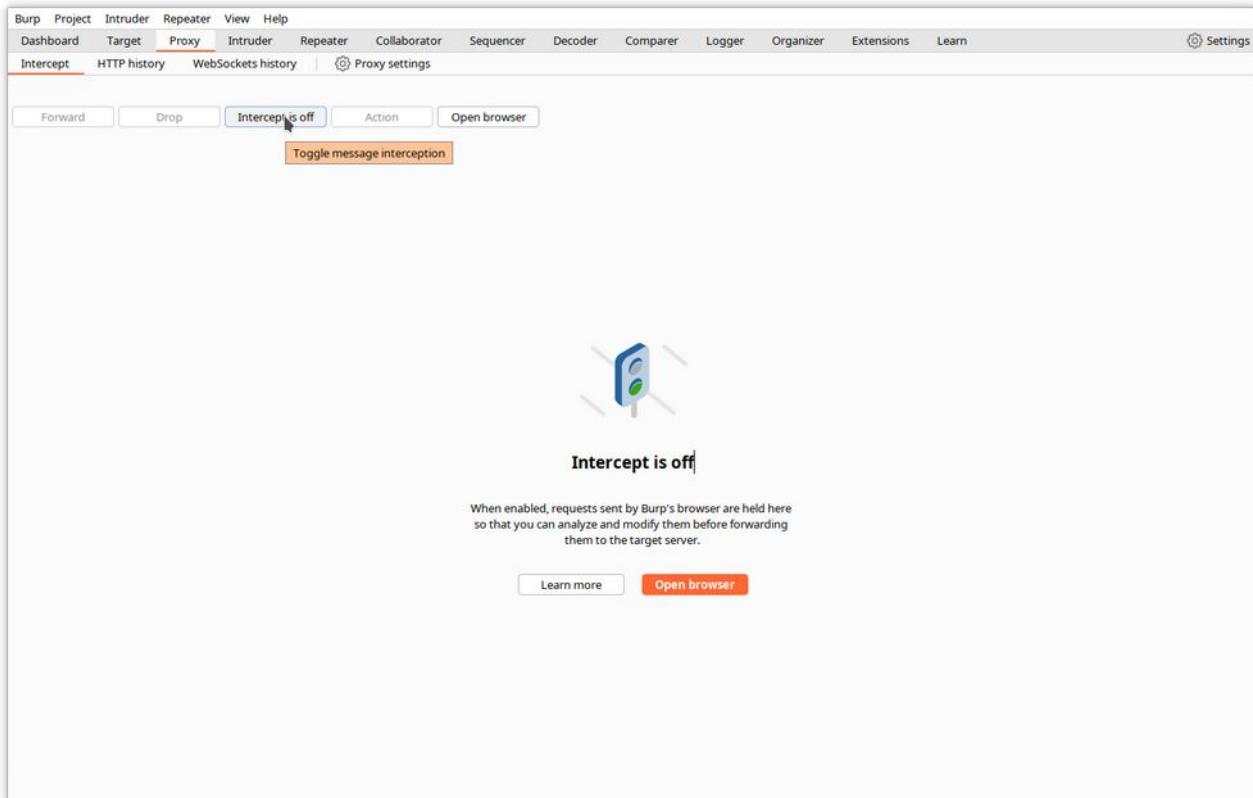
Let's first execute a transaction on a basic website that simply asks for your name and greets you by your name.



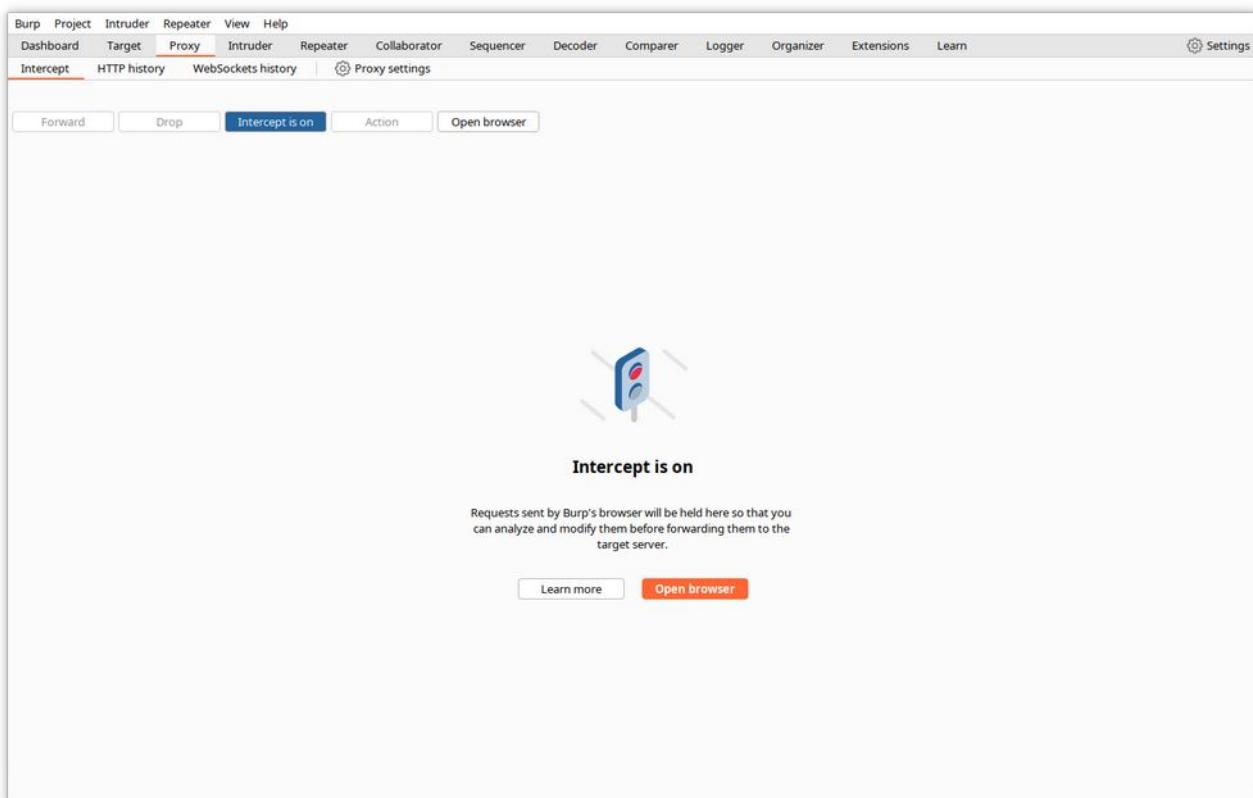
We will now modify the data sent by the POST request.

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.

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

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

Forward Drop Intercept is on Action Open browser

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
```

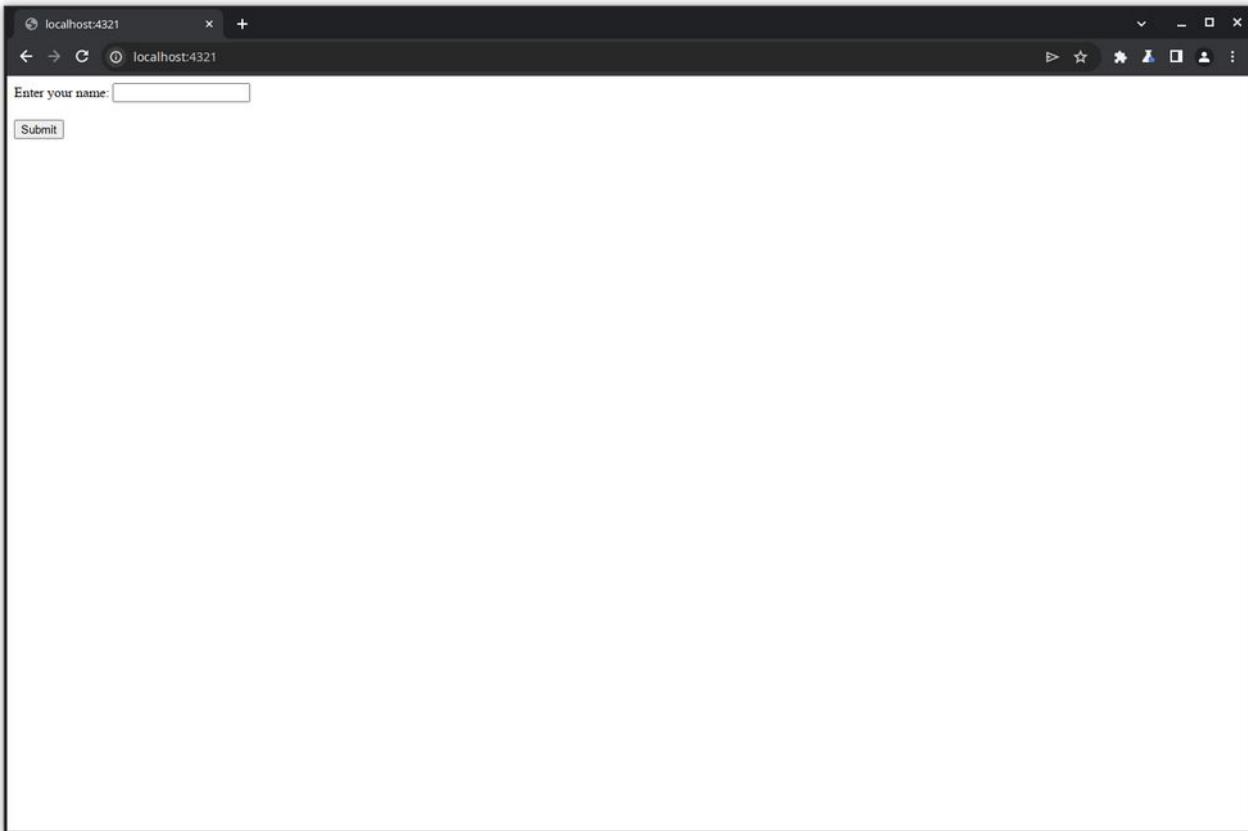
Add notes HTTP/1

Inspector Request attributes 2 Request query parameters 0 Request body parameters 0 Request cookies 0 Request headers 15 Notes

0 highlights

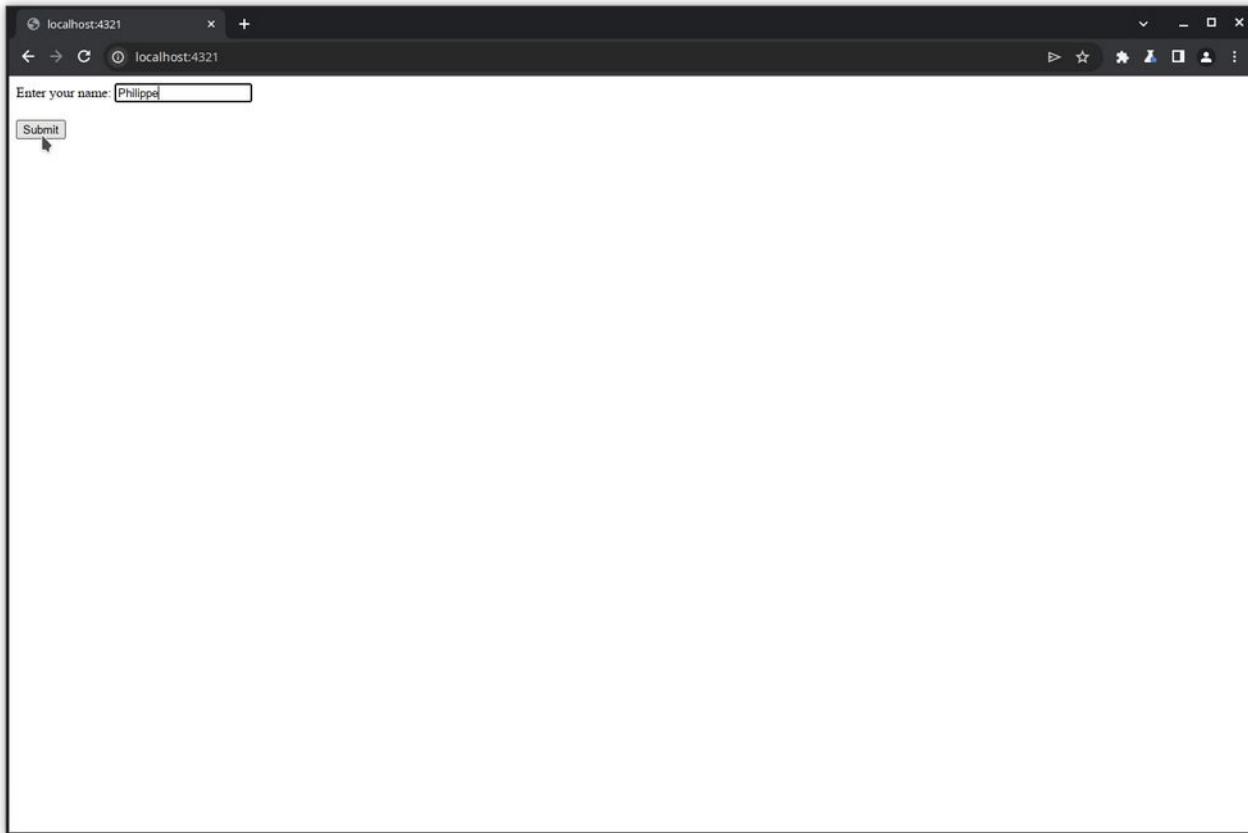
Click **Forward** in the Burp window to forward the request to the website.

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.



A screenshot of the Burp Suite proxy tool. The "Proxy" tab is selected. In the "HTTP history" section, there is one captured item: a GET request to "http://localhost:4321" with a status code of 200, length of 521, and MIME type of HTML. The "Request" pane shows the raw HTTP request, which includes headers like Host, User-Agent, and Accept, and a body containing a form with a username input field. The "Response" pane shows the raw HTML response, which contains the form with the same fields. The "Inspector" pane on the right shows the request attributes, request headers (15 items), and response headers (3 items). The "Notes" pane is empty.

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



The request is sent to Burp.

A screenshot of the Burp Suite interface, specifically the "Proxy" tab. The "Selected text" panel on the right side of the interface has the word "Philippe" highlighted. The "Raw" tab of the proxy view shows a captured POST request. The request body contains the key-value pair "username=Philippe". The "Inspector" panel on the right shows various details about the selected text, such as "Decoded from: URL encoding" and "Philippe".

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

The screenshot shows the Burp Suite interface in the Proxy tab. A request to `http://localhost:4321 [127.0.0.1]` is selected. The 'Raw' tab is active, displaying the following modified HTTP 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;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[
```

The 'Inspector' panel on the right shows the selected text 'Gaston' has been decoded from URL encoding. The 'Notes' panel is also visible.

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

The screenshot shows a browser window with the address bar set to `localhost:4321`. The page content displays the modified response: **Hello, Gaston!**

The screenshot shows the NetworkMiner interface with the following details:

- Project:** Project 1
- Proxy:** Target → Localhost
- Intercept:** Enabled
- HTTP history:** 2 items
- Selected item:** POST / from localhost:4321
- Request Headers:**
  - Host: localhost:4321
  - Content-Length: 17
  - Cache-Control: max-age=0
  - sec-ch-ua: "Chromium";v="115", "Not?A\_Brand";v="24"
  - sec-ch-ua-mobile: ?0
  - sec-ch-ua-platform: "Linux"
  - Upgrade-Insecure-Requests: 1
  - Origin: http://localhost:4321
  - 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: \*/\*
  - Sec-Fetch-Site: same-origin
  - Sec-Fetch-Mode: navigate
  - Sec-Fetch-User: ?1
  - Sec-Fetch-Dest: document
  - Referer: http://localhost:4321/
  - Accept-Encoding: gzip, deflate, br
  - Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
  - Connection: close
- Request Body:** username=Philippe
- Response Headers:**
  - HTTP/1.0 200 OK
  - Server: BaseHTTP/0.6 Python/3.8.10
  - Date: Wed, 02 Oct 2024 14:29:59 GMT
  - Content-type: text/html
- Response Body:** Hello, Gaston!
- Inspector:** Shows Request attributes (2), Request body parameters (1), Request headers (19), Response headers (3), 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.

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 button in the top right corner says "LAB Not solved". The main content area has a banner with the text "WE LIKE TO SHOP" and a stylized arm icon. Below the banner are four product cards:

- Lightweight "I33t" Leather Jacket**: Price \$1337.00
- The Alternative Christmas Tree**: Price \$61.56
- Eye Projectors**: Price \$5.70
- Caution Sign**: Price \$64.57

Below the products is a row of three images: a person climbing a skyscraper, a bunch of colorful butterfly wings, and a man wearing a fedora hat.

## Step 2: Log in

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

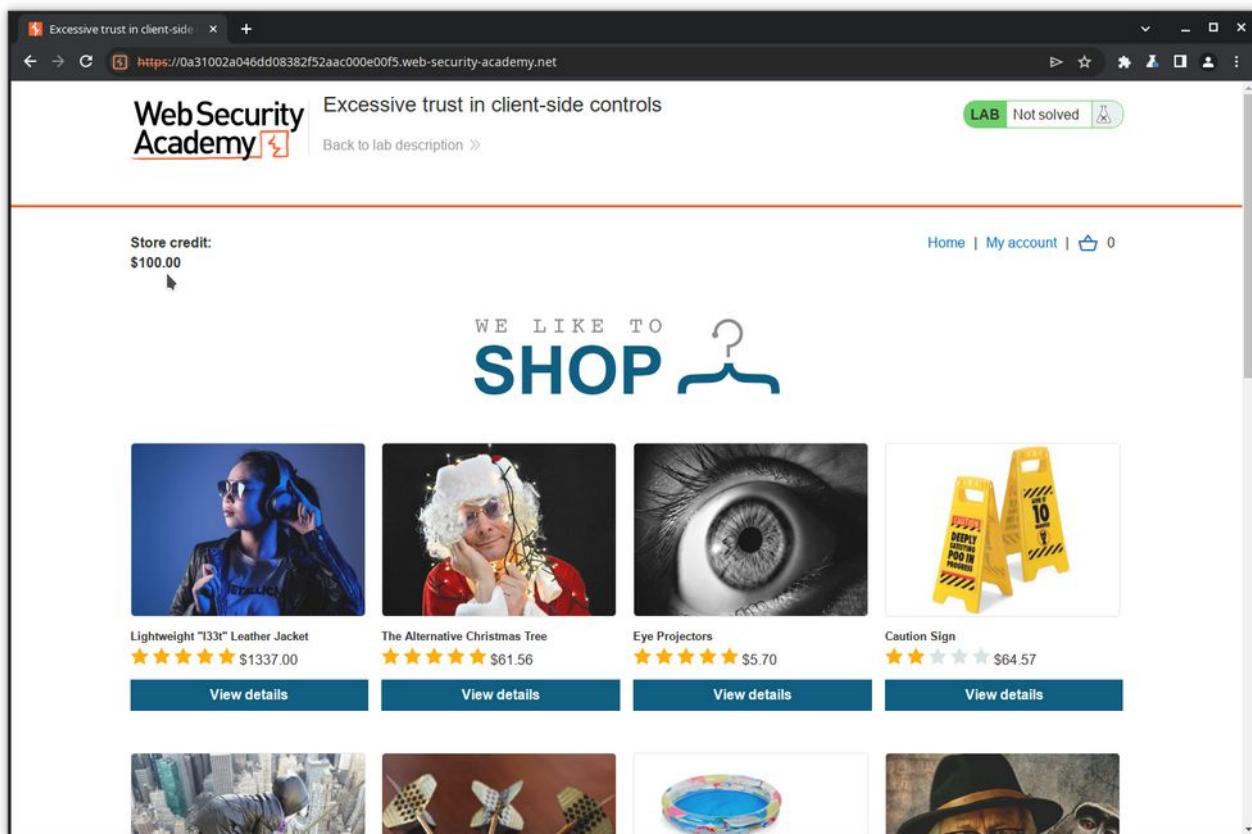
The screenshot shows a web browser window for the 'Excessive trust in client-side controls' lab on Web Security Academy. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/login>. The page title is 'Excessive trust in client-side controls'. At the top right, there is a green 'LAB' button, a 'Not solved' status, and a test tube icon. Below the title, there is a link 'Back to lab description >'. On the right side of the header, there are links for 'Home', 'My account', and a shopping cart icon showing '0'. The main content area is titled 'Login'. It contains two input fields: 'Username' with 'wiener' typed in and 'Password' with '.....'. A green 'Log in' button is at the bottom. A mouse cursor is hovering over the 'Log in' button.

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

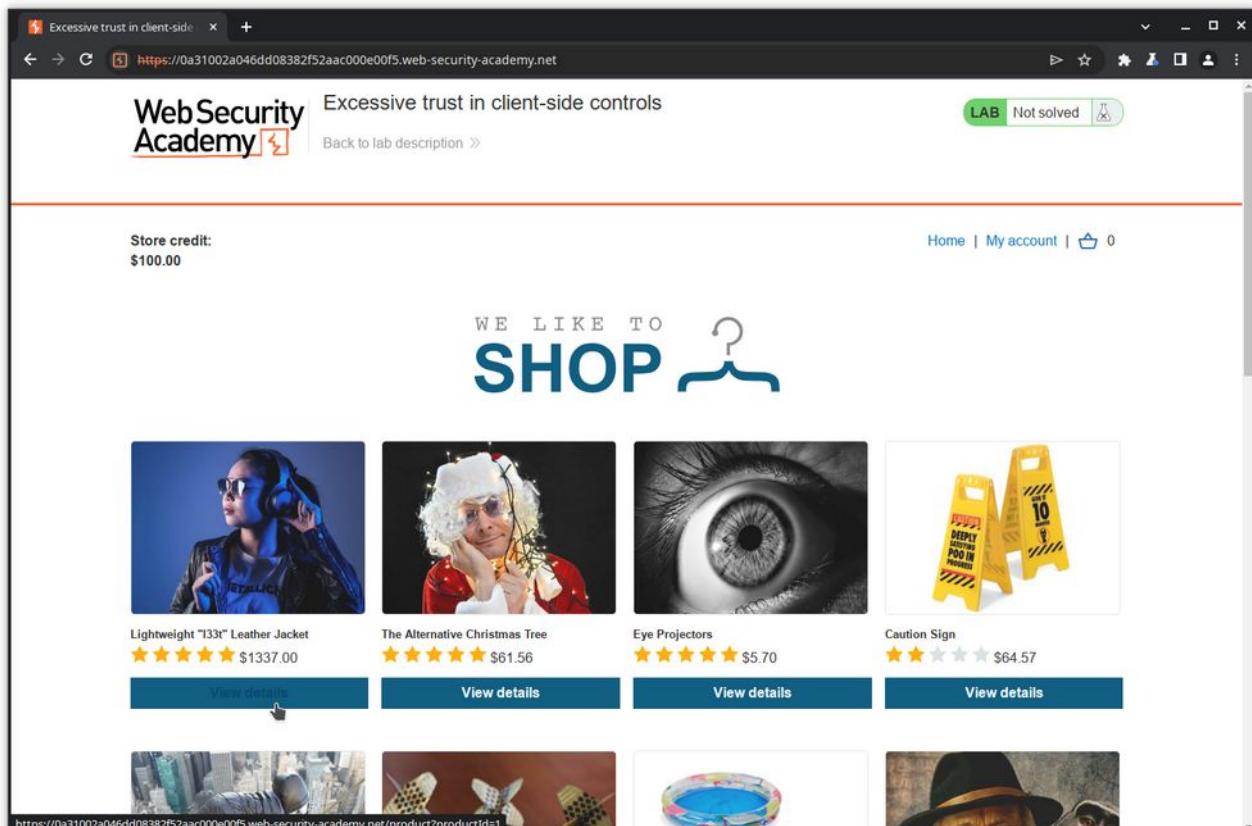
The screenshot shows a web browser window for the 'Excessive trust in client-side controls' lab on Web Security Academy. The URL is <https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/my-account?id=wiener>. The page title is 'Excessive trust in client-side controls'. At the top right, 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 right side of the header, there are links for 'Home', 'My account', a shopping cart icon showing '0', and 'Log out'. The main content area is titled 'My Account'. It displays 'Store credit: \$100.00'. Below this, it says 'Your username is: wiener'. There is an input field for 'Email' with a placeholder 'Email' and a green 'Update email' button. A mouse cursor is hovering over the 'Log out' link.

### 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.

Excessive trust in client-side controls

Back to lab description >

LAB Not solved

Store credit:  
\$100.00

Cart

Name	Price	Quantity
Lightweight "I33t" Leather Jacket	\$1337.00	1

Coupon:  
Add coupon

Apply

Total: \$1337.00

Place order

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

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 "I33t" Leather Jacket	\$1337.00	1

Coupon:  
Add coupon

Apply

Total: \$1337.00

Place order

## 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 'Proxy' tab selected. The 'HTTP history' tab is active. The history list displays several requests and responses. One specific POST request to '/cart' with parameter 'productID=1' is highlighted. The response body contains HTML code for a purchase confirmation page, including hidden fields for product ID and quantity.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
144	https://0a31002a046dd08382f5...	POST	/cart		✓	302	85			Excessive trust in client...	✓	34.246.129.62	
143	https://0a31002a046dd08382f5...	GET	/cart?err=INSUFFICIENT_FUNDS		✓	200	6523	HTML				✓	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	

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

Let's check the POST action.

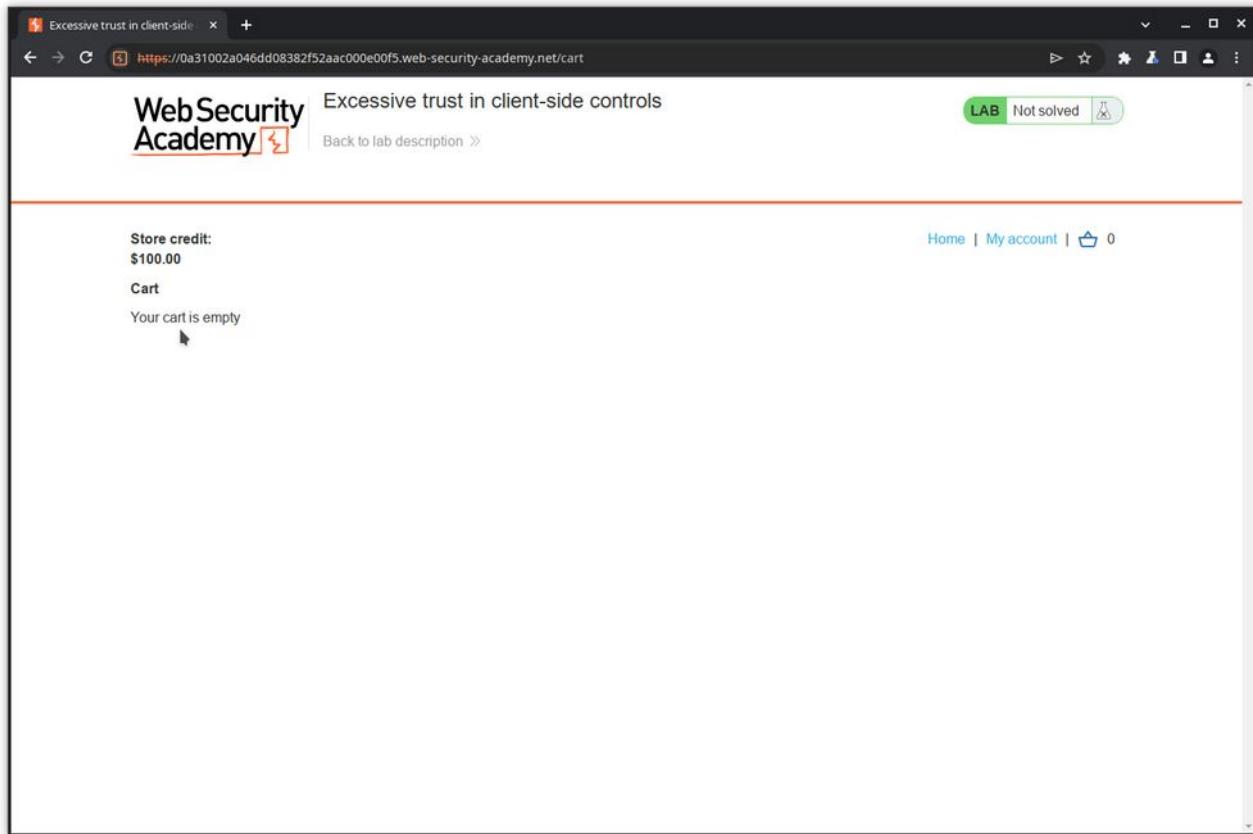
Screenshot of the Burp Suite interface showing a list of captured requests. The selected request is a POST to '/cart' with parameters productId=1, quantity=1, and price=133700. The response shows a 302 Found status with a redirect to '/product?productId=1'. The Inspector panel shows the selected text: 'productId=1&redirect=PRODUCT&quantity=1&price=133700'.

#	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

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.

Screenshot of a browser window showing the 'Web Security Academy' website. The page title is 'Excessive trust in client-side controls'. The URL is 'https://0a31002a046dd08382f52aac000e00f5.web-security-academy.net/cart?err=INSUFFICIENT\_FUNDS'. The page displays a message: 'Not enough store credit for this purchase'. Below this, there is a table for the 'Cart' containing one item: 'Lightweight "I33l" Leather Jacket' at '\$1337.00'. The 'Quantity' is set to '1'. To the right of the quantity is a 'Remove' button, which is being clicked. A tooltip 'LAB Not solved' is visible in the top right corner.

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 responses. A context menu is open over a selected POST request, with the option 'Send to Repeater' highlighted. The right pane contains the 'Inspector' tool, which shows the 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 POST message. The right pane contains the 'Inspector' tool, which shows the 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. The Request pane displays a POST /cart HTTP/2 message with various headers and a body containing the URL-encoded parameters: productId=1&redirect=PRODUCT&quantity=1&price=1. The Response pane shows the server's response, which is a 302 Found status code with a Location header pointing back to the same URL. The Inspector pane on the right highlights the selected text in the request body.

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

This screenshot is identical to the one above, showing the Repeater tab with the same request, response, and inspector details. The status bar at the bottom right indicates a total size of 100 bytes and a duration of 230 milliseconds.

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

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

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 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.

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

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>.

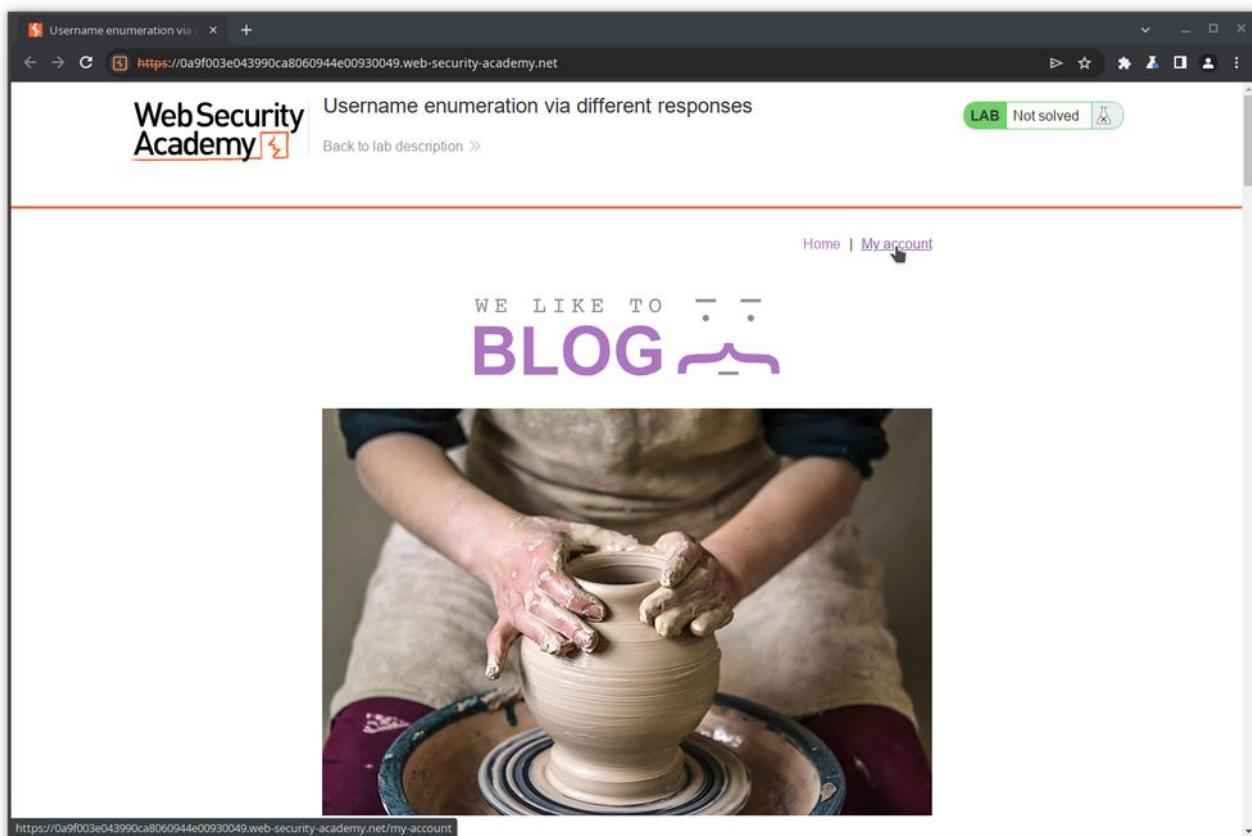
Burp Intruder is a tool that allows sending the same HTTP request several times, inserting different payloads into predefined positions each time. In this example, we will use it to brute force the username and password to get access to a web application.

### Step 1: Access the application

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

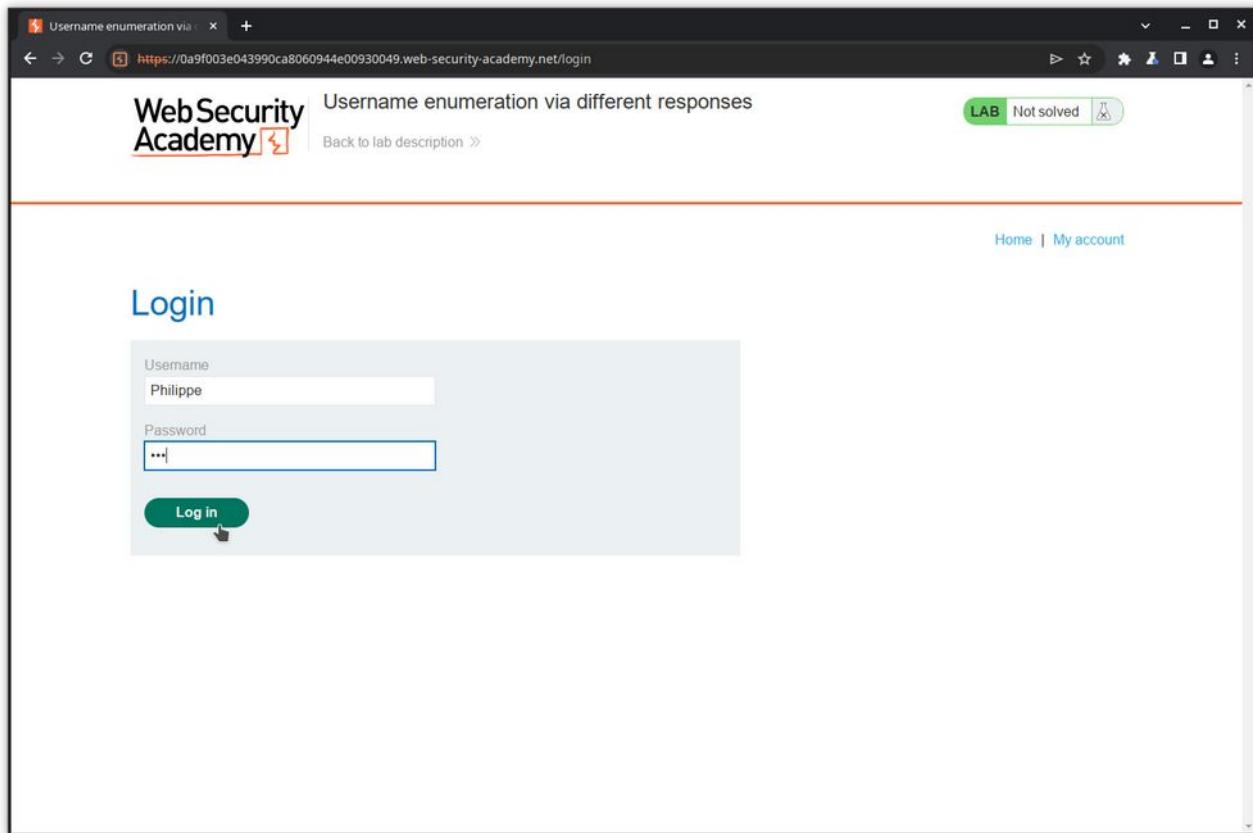
### Step 2: Try to log in

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

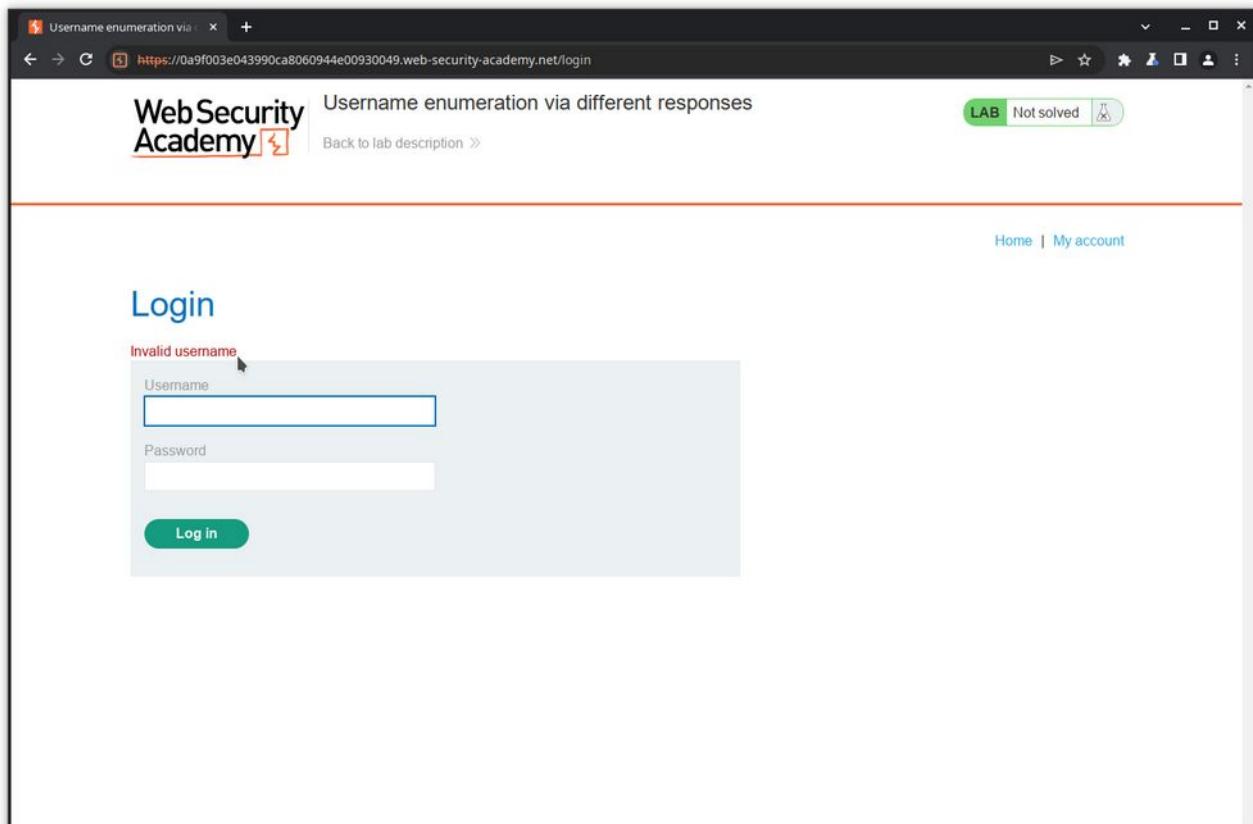


The login page opens. As we don't have valid credentials, we will try with a random username and password and observe the content of the POST request and the response.

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 with the 'Proxy' tab selected. The 'HTTP history' section displays several network requests. The last request is a POST to '/login'. A context menu is open over this 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 single request is displayed with a payload inserted at position 1. The payload is a simple list containing 'abc'. The 'Attack type' is set to 'Sniper'. The 'Payload positions' section shows one payload position has been added. The status bar indicates 1 highlight and a length of 1030 bytes.

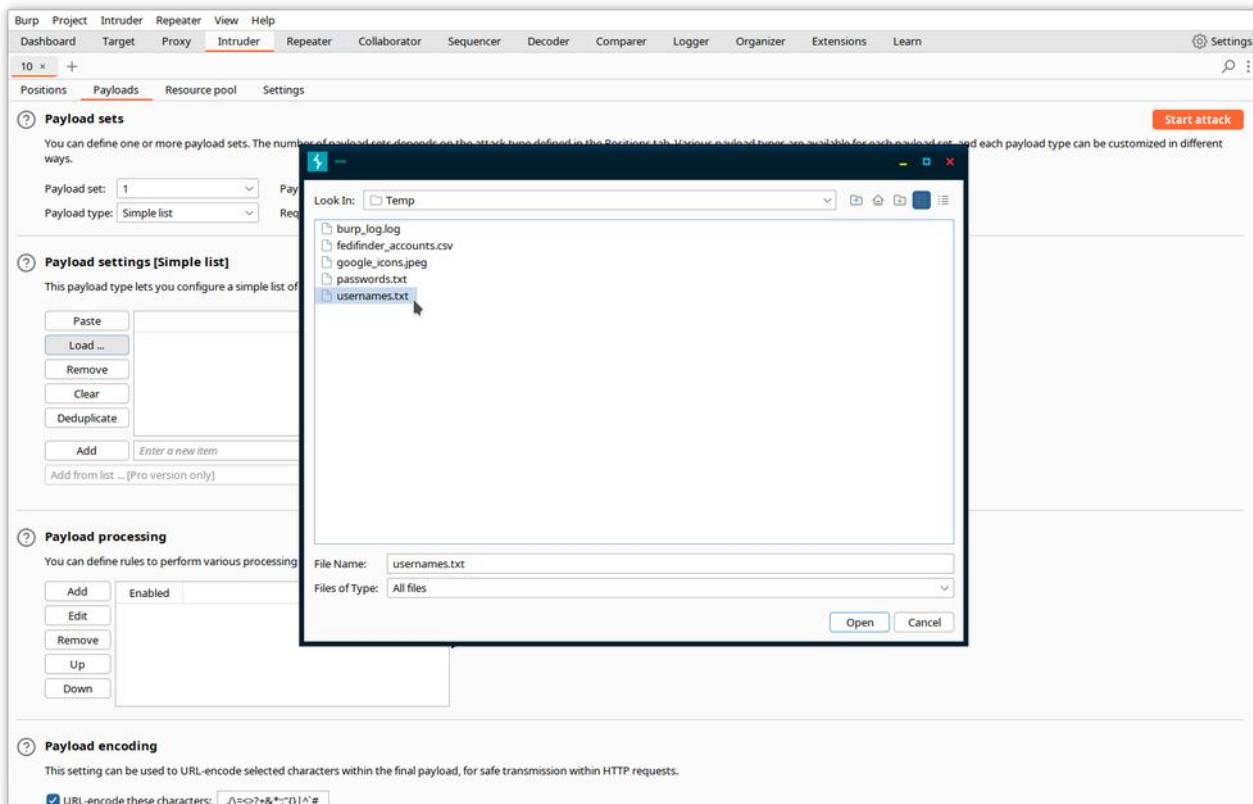
Check the **Attack type** at the top of the screen is set to **Sniper**. Sniper attack inserts a single set of payload, one at a time, into one or more positions in the request.

#### Step 4: Add the payload

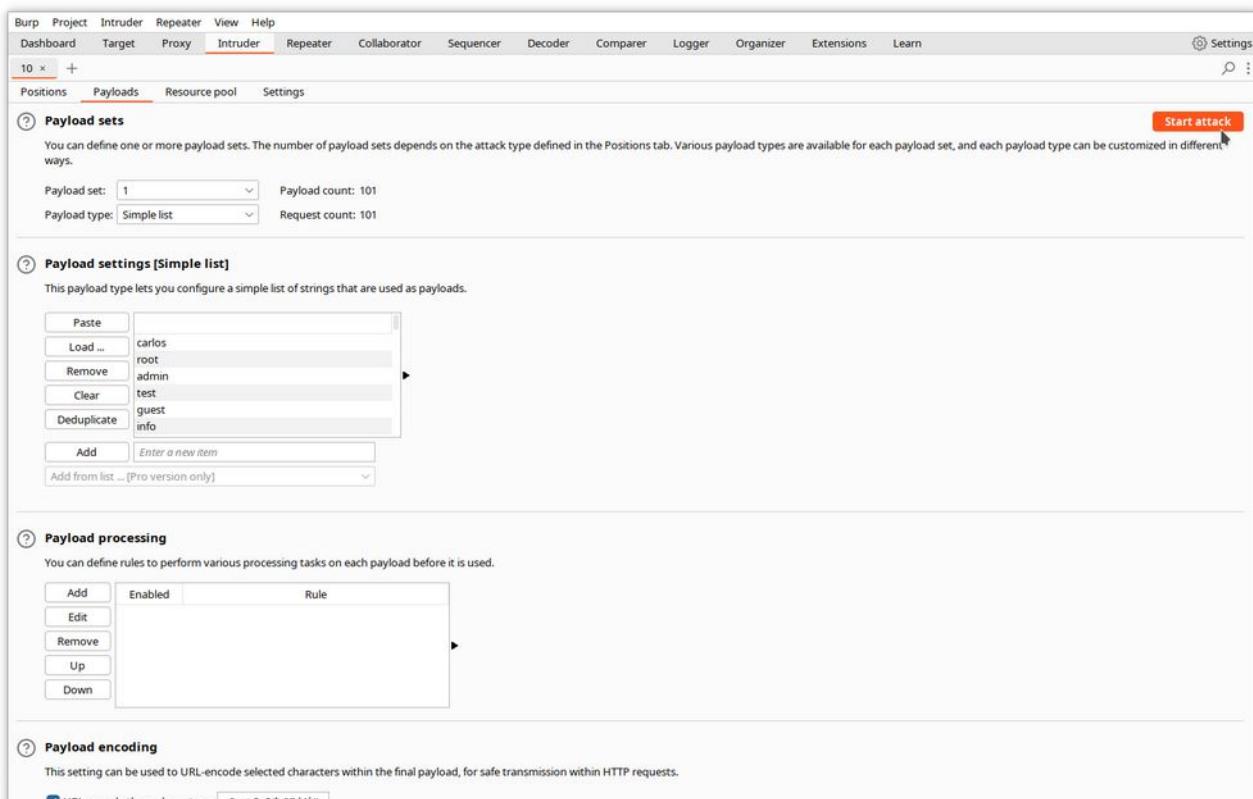
In Intruder, click the **Payloads** tab. This window allows you to input a list of values for the payload, or to specify a file containing this list.

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. The 'Payload sets' section shows one set with a payload count of 0. The 'Payload settings [Simple list]' section shows a list of items: 'Load items from file' (which is highlighted), 'Paste', 'Load ...', 'Remove', 'Clear', 'Deduplicate', 'Add' (with an input field 'Enter a new item'), and 'Add from list ... [Pro version only]'. The 'Payload processing' section shows a table with columns 'Add', 'Enabled', and 'Rule'. The 'Payload encoding' section contains a checkbox for URL-encoding specific 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.

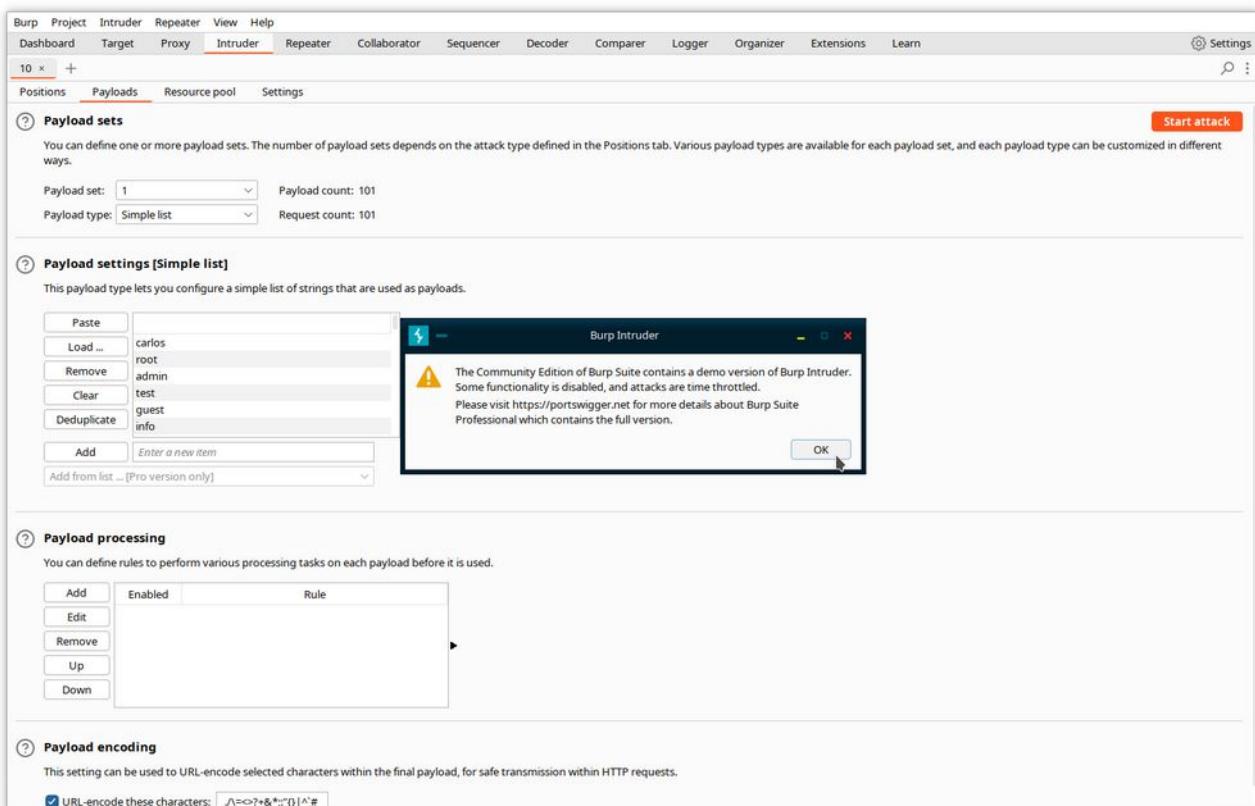


Note that the **Payload count** and **Request count** values are updated with the number of entries in the list.



## 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 main pane, there's a section for 'Payload sets' with settings for a payload count of 101 and a request count of 101. Below this is a 'Payload settings [Simple list]' section containing a list of items: carlos, root, admin, test, guest, and info. An 'Add' button is available to enter new items. 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 tabs like 'Proxy' and 'Repeater' are visible at the top.

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', 'Columns', 'Results' (which is selected), 'Positions', 'Payloads', 'Resource pool', and 'Settings'. Below this is a search bar with the placeholder 'Filter: Showing all items'. A table lists 10 requests, each with a number, payload, status code, error, timeout, length, and comment. The first request (adserver) has a blue background. The 'Request' tab is selected in the main view, showing a detailed log of the request. The log includes:

```
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
```

Below the log, there are buttons for '?', 'gear', 'back', 'forward', 'Search', and a search icon. A progress bar at the bottom indicates the task is '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 them, a navigation bar includes 'Results' (which is selected), 'Positions', 'Payloads', 'Resource pool', and 'Settings'. A search bar with the placeholder 'Filter: Showing all items' is present. The main area displays a table of requests:

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	

Below the table, a section titled 'Response' is expanded. It shows a 'Pretty' view of the HTML response with line numbers 47 through 57. The line containing the error message 'Incorrect password' is highlighted. The response content includes:

```
47     </section>
48     </header>
49     <header class="notification-header">
50     </header>
51     <h1>
52     Login
53     </h1>
54     <section>
55     <p class=is-warning>
56     Incorrect password</p>
57     </section>
<form class=login-form method=POST action="/login">
  <label>
    Username
  </label>
  <input required type=username name=username autofocus>
  <label>
    Password
  </label>
```

At the bottom of the response view, there are icons for help, settings, back, forward, and search, along with a status bar indicating '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 table of requests with columns for Request, Payload, Status code, Error, Timeout, Length, and Comment. Below the table is a list of payloads: adserver, carlos, root, admin, test, guest, info, mysql, and user. A modal dialog titled 'Do you want to save this attack?' is open, asking if the user wants to keep it in memory or discard it. The 'Discard' button is highlighted with a red box.

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 messages. A context menu is open over a selected response message, with 'Send to Intruder' highlighted with a red box.

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 selection dialog is open, showing files in the 'Temp' directory, with 'passwords.txt' selected. The 'Open selected file' button is highlighted with an orange box.

Then, start the new attack.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payload count' is set to 100. The 'Payload type' is set to 'Simple list'. The 'Payload settings [Simple list]' section shows a list of payloads including '123456', 'password', '12345678', 'qwerty', '123456789', and '12345'. The 'Payload processing' section shows a table with one row. 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
```

?

Search

0 highlights

Finished

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

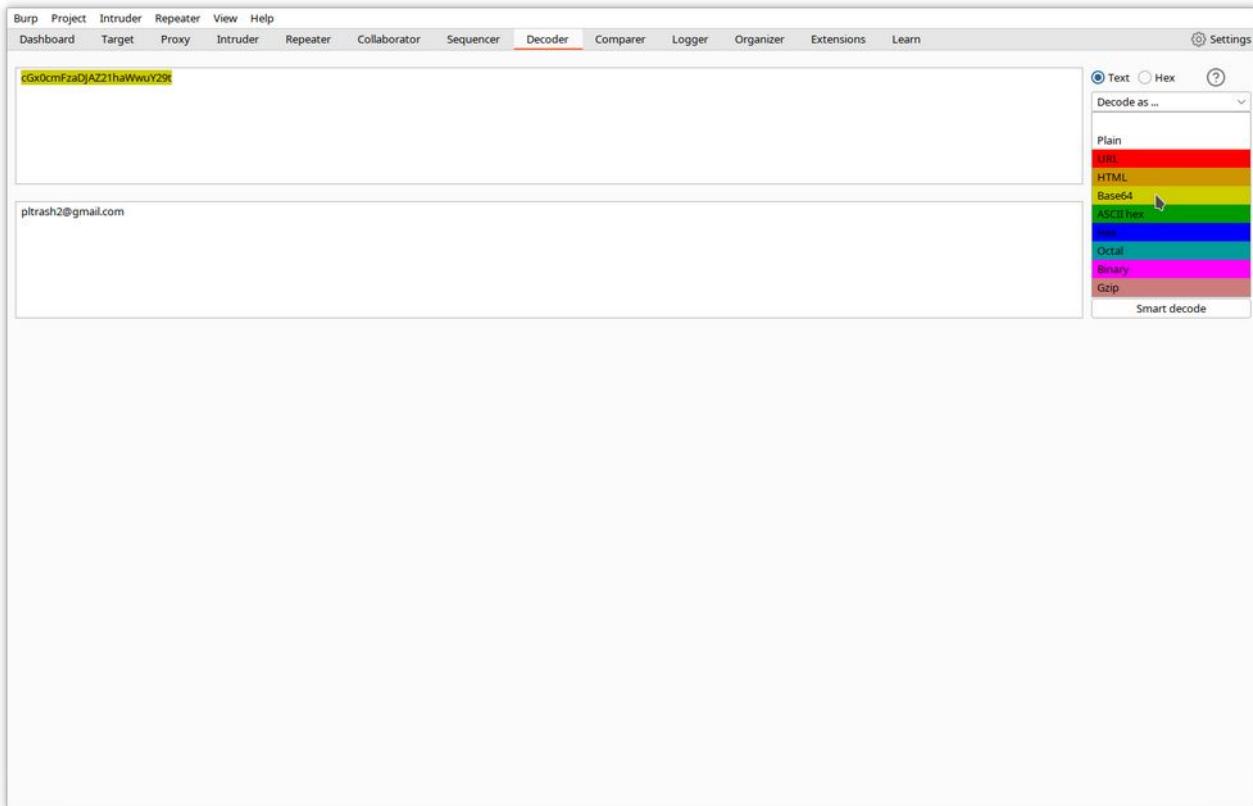
Let's confirm in the browser by performing an interactive login with the found credentials.

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

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:**

Select item 2:	#	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...

Key: Modified Deleted Added Sync views

Compare ... Words Perform a word-level compar