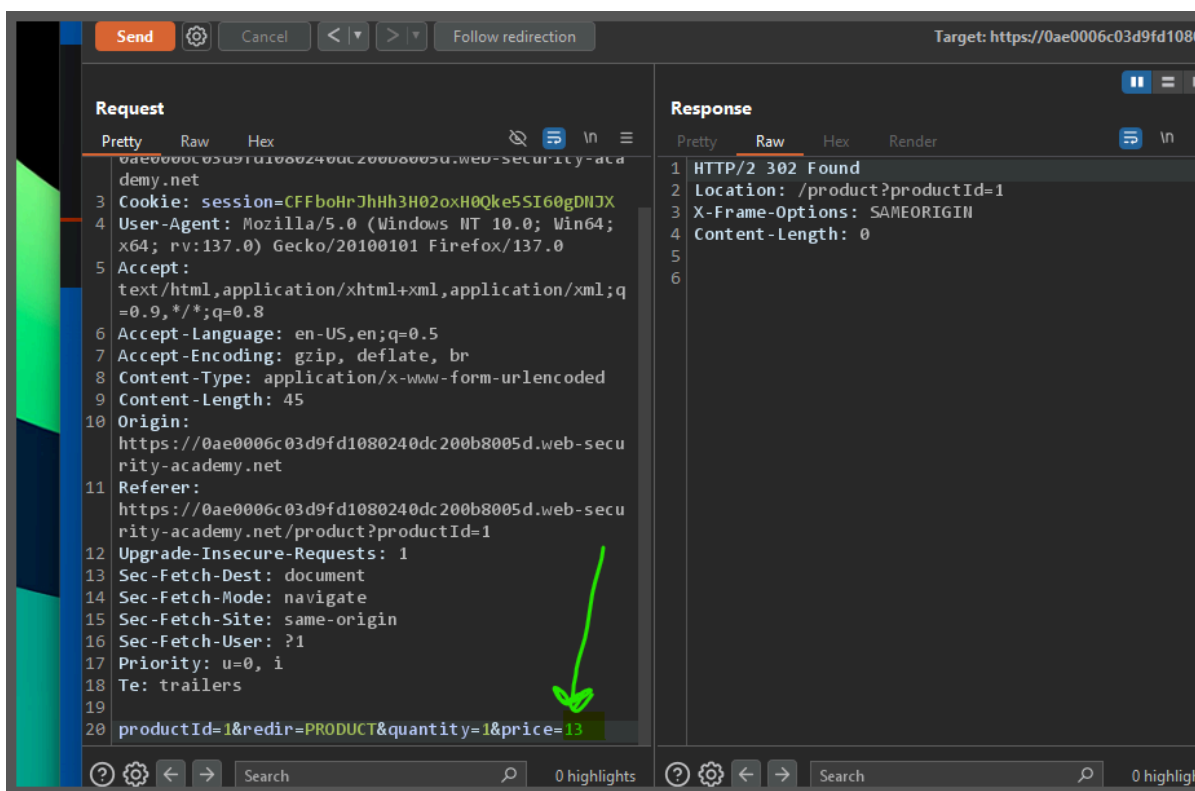


Business Logic flaw

Lab: Excessive trust in client-side controls

This lab doesn't adequately validate user input. You can exploit a logic flaw in its purchasing workflow to buy items for an unintended price. To solve the lab, buy a "Lightweight I33t leather jacket".

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



Lab: Inconsistent handling of exceptional input

This lab doesn't adequately validate user input. You can exploit a logic flaw in its account registration process to gain access to administrative functionality. To solve the lab, access the admin panel and delete the user

`carlos`.

do directory busting

The screenshot shows a web browser's developer tools interface. On the left, the 'Site map filter' shows a directory listing for the URL `https://0a7c0052038dd78782db513300b000cc...`. The directory structure includes `/`, `academyLabHeader`, `admin` (highlighted), `admin`, `favicon.ico`, `image`, `login`, `logout`, `my-account` (highlighted), `my-account`, `product`, `product`, `register`, `register`, and `resources`. The main panel displays the response for the `GET /admin` request, which is an HTML document with a status code of 401. The response content is an SVG image with a back arrow, rendered in a dark theme.

The screenshot shows a Web Security Academy lab page titled 'Inconsistent handling of exceptional input'. The lab is marked as 'Solved' and 'LAB'. A large orange banner with a green 'X' over it says 'Congratulations, you solved the lab!'. Below the banner, there are links for 'Share your skills!', 'Continue learning >>', and a navigation bar with 'Home', 'My account', and 'Register'. A green box highlights the text 'Admin interface only available if logged in as a DontWannaCry user'.

The screenshot shows a Sublime Text window titled "\\\wsl.localhost\kali-linux\home\charon\temp\exploit.php - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tab bar shows three files: burp= @SJW5]gX35[q9:=6EWEN7/bk\$@/[GxSo9, session.php, and exploit.php. The main editing area displays the following code:

```
1 |  
2 |  
3 | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
   .exploit-0ac600220306d7d6828950570184000a.exploit-server.net
```

A green arrow originates from the handwritten text "255 Char" at the bottom right and points to the final character of the first line of 'a's.

[illegible]

[Home](#)

Register

If you work for DontWannaCry, please use your @dontwannacry.com email address

Username

test3

Email

iaaaa@dontwannacry.com.exploit-0ac600220306d7d6828950570184000a.exploit-server.net

Password

••••••••

Register

get the email on your email client
login

now you see that update email
do : test@dontwannacry.com
now your admin
delete the carlos

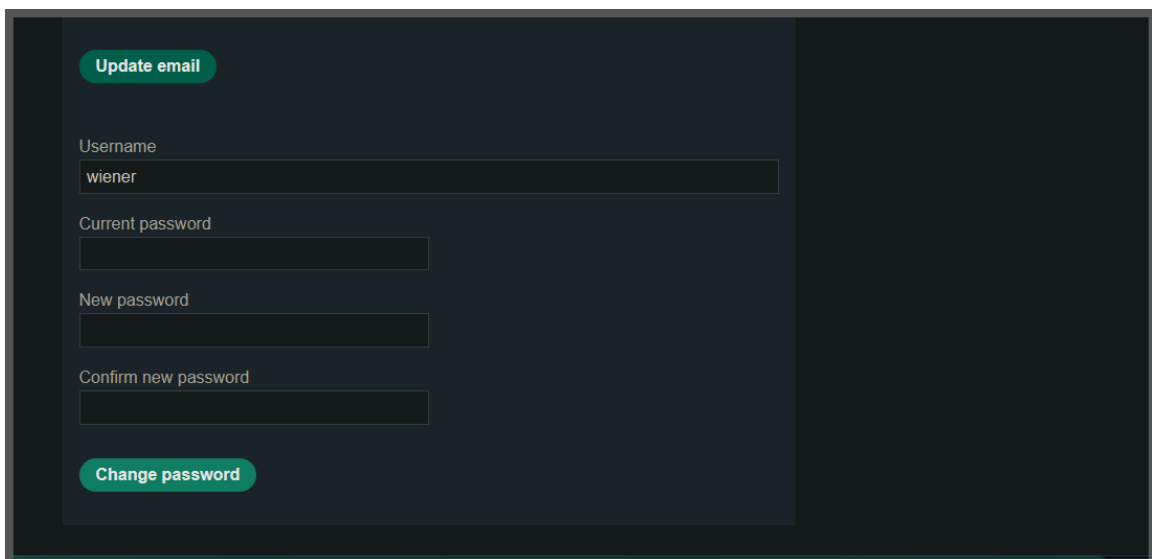
Lab: Weak isolation on dual-use endpoint

This lab makes a flawed assumption about the user's privilege level based on their input. As a result, you can exploit the logic of its account management features to gain access to arbitrary users' accounts. To solve the lab, access the

`administrator` account and delete the user `carlos`.

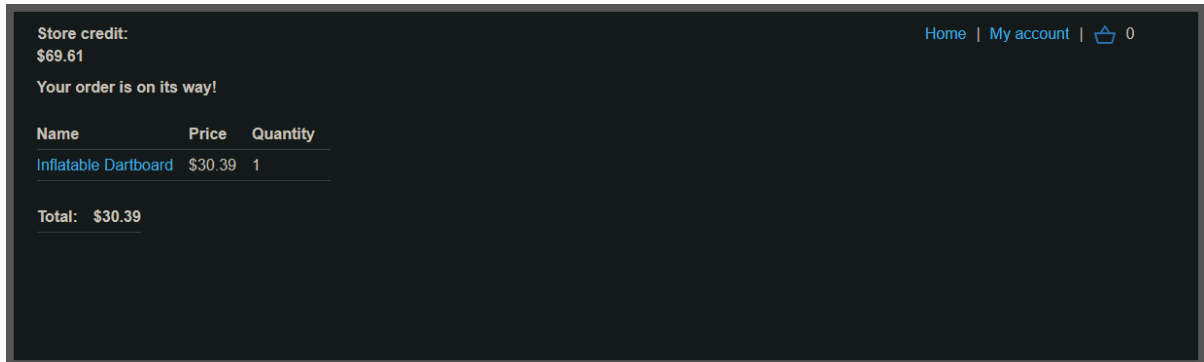
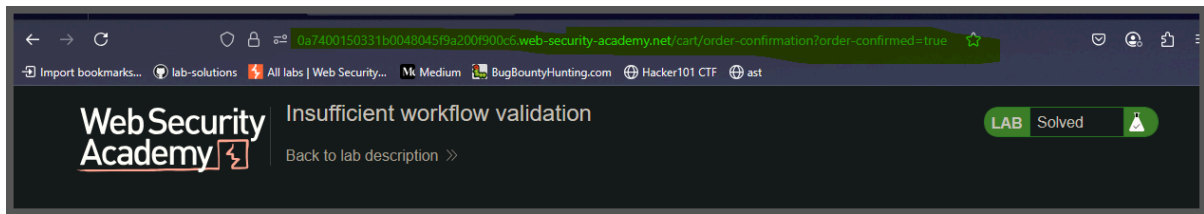
You can log in to your own account using the following credentials:

`wiener:peter`

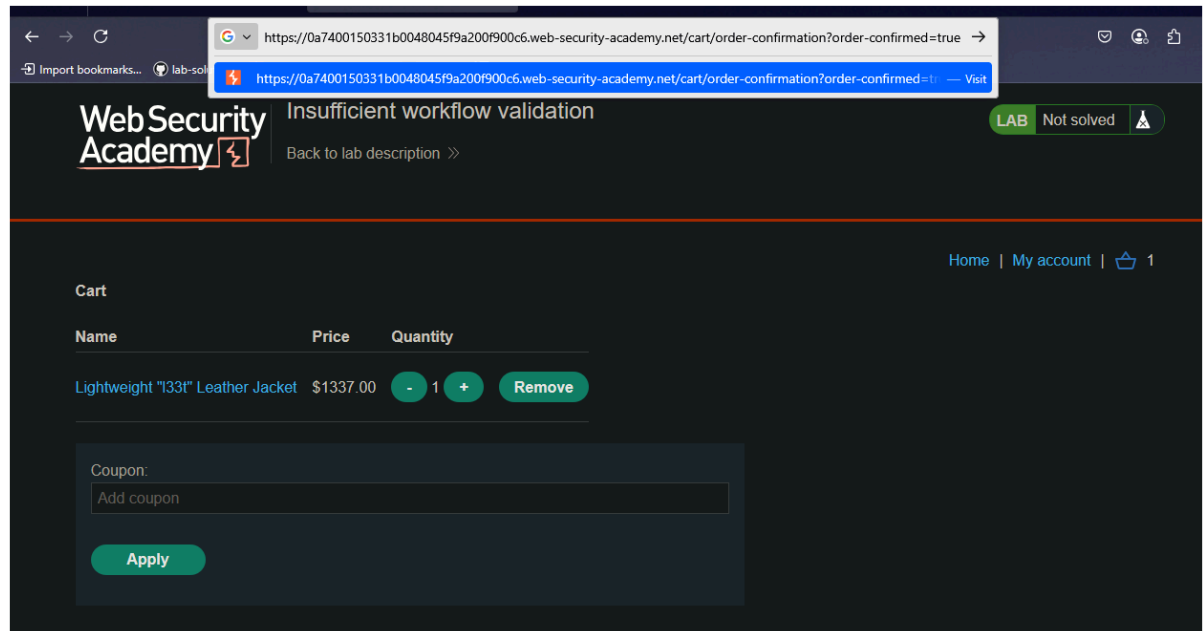


The screenshot shows a dark-themed web interface for account management. At the top left, there is a green button labeled "Update email". Below it, the "Username" field contains the text "wiener". The "Current password" field is empty. The "New password" field is empty. The "Confirm new password" field is empty. At the bottom left, there is a green button labeled "Change password".

once we place the order is shifted to this url



so now add the leather jacket and put this url



solved the lab

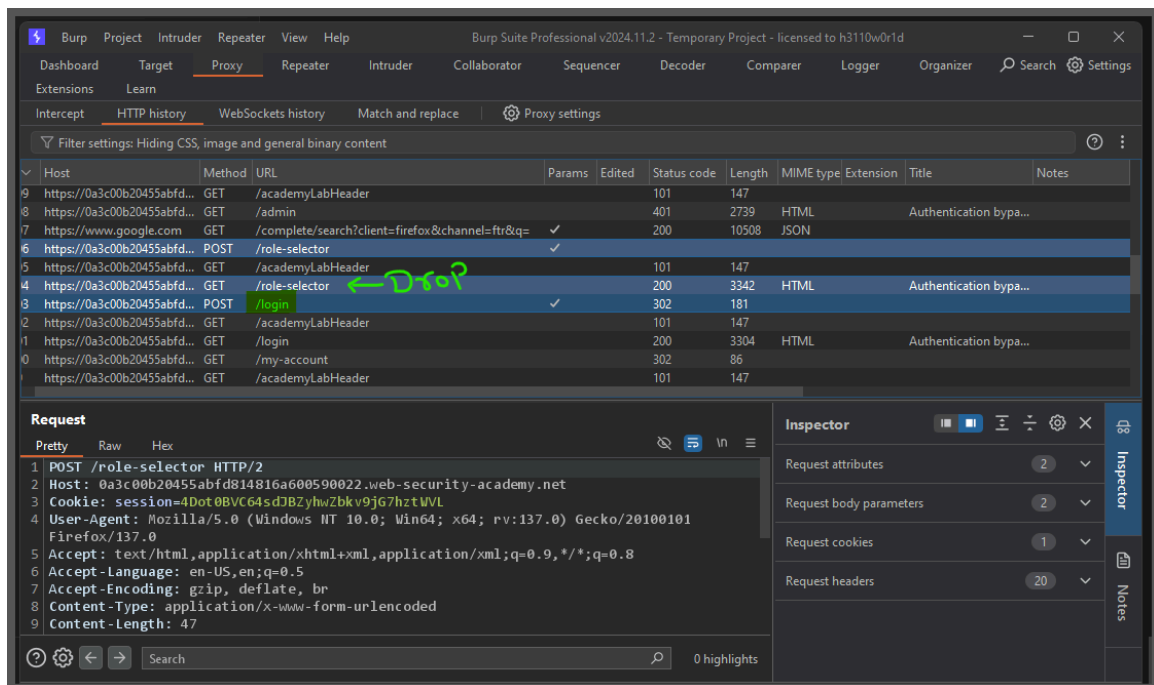
Lab: Authentication bypass via flawed state machine

This lab makes flawed assumptions about the sequence of events in the login process. To solve the lab, exploit this flaw to bypass the lab's authentication, access the admin interface, and delete the user

carlos .

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

wiener:peter



Turn on intercept

make login

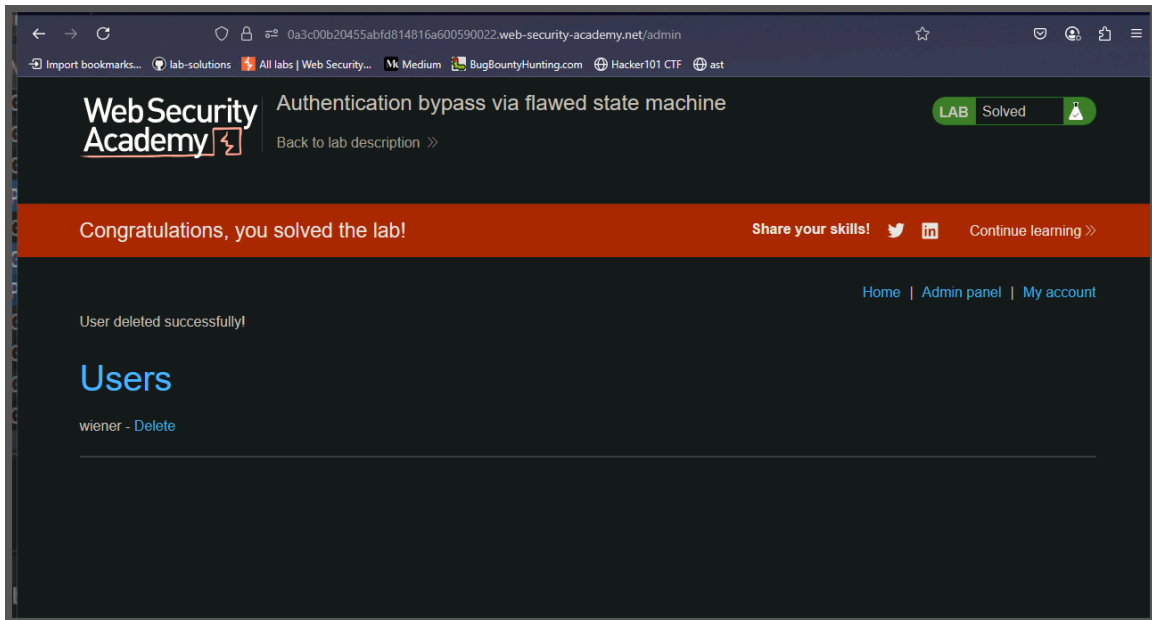
now you see next request is to role selector

drop that request

turn off intercept

and remove the rolesselector from url

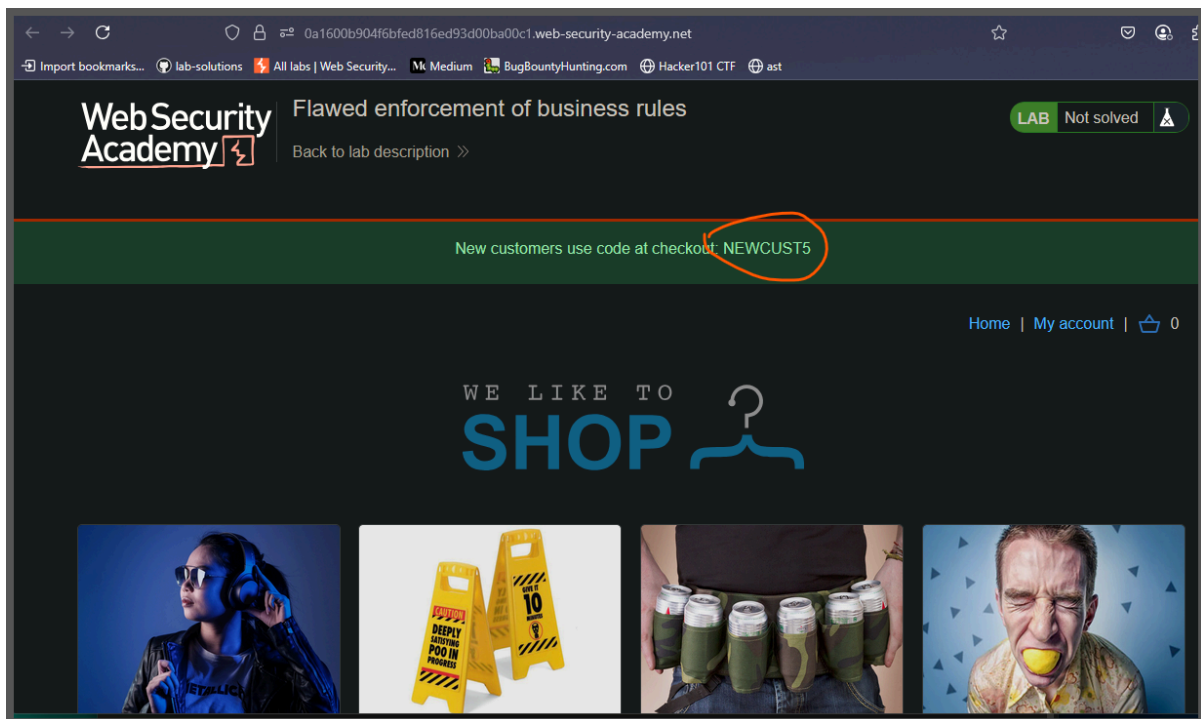
now go your admin



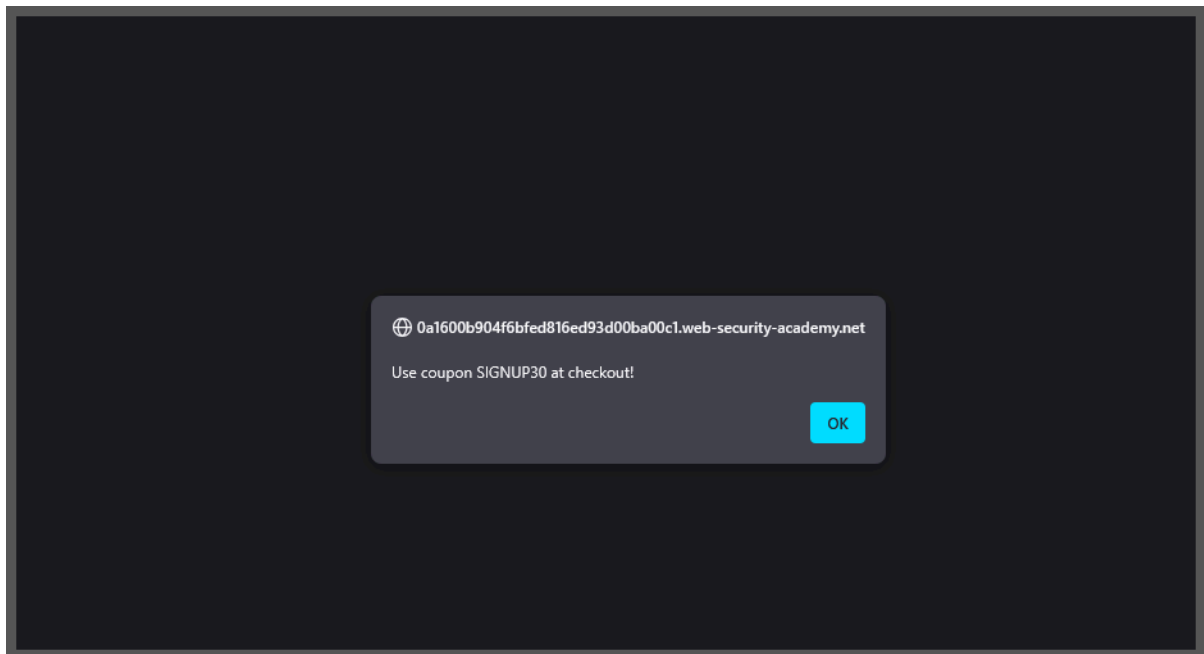
Lab: Flawed enforcement of business rules

This lab has a logic flaw in its purchasing workflow. To solve the lab, exploit this flaw to buy a "Lightweight I33t leather jacket".

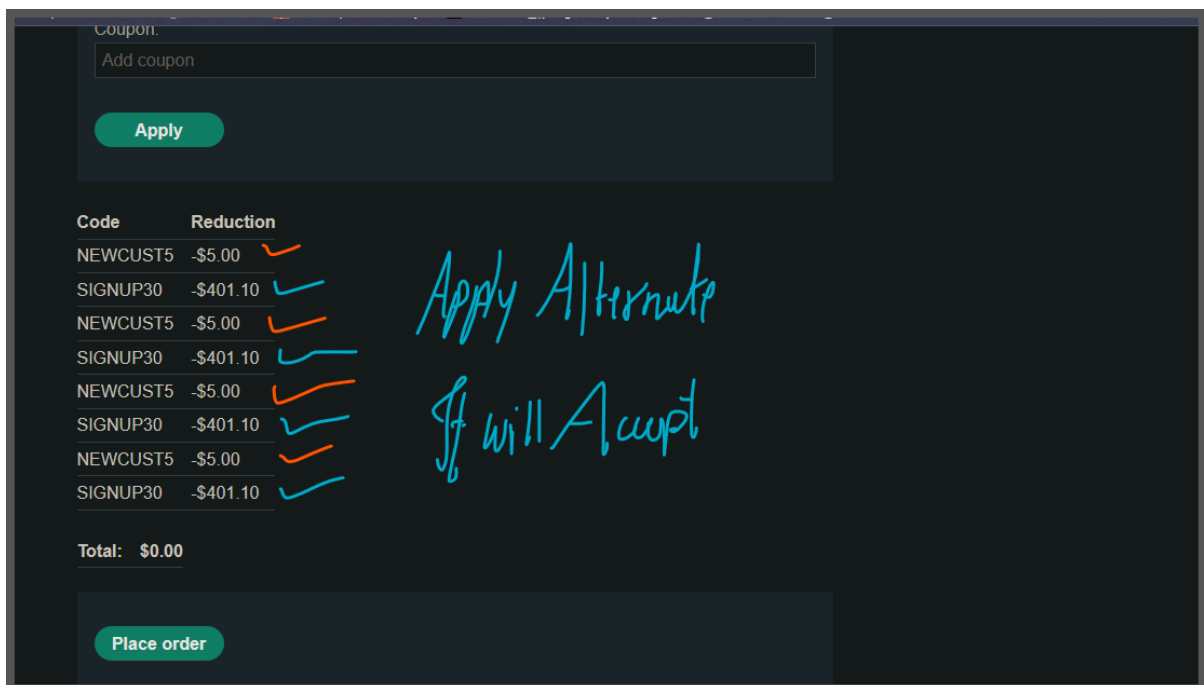
You can log in to your own account using the following credentials: `wiener:peter`



scroll down you will see sighup of newlatter make it



till now we got two cupons



Lab: Infinite money logic flaw

This lab has a logic flaw in its purchasing workflow. To solve the lab, exploit this flaw to buy a "Lightweight I33t leather jacket".

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

lab is easy but there are few more steps
ther is and logic vulnerability

using cupon get discount
buy the product and hence due to discont you get few money back
repeating this you will create infinate money

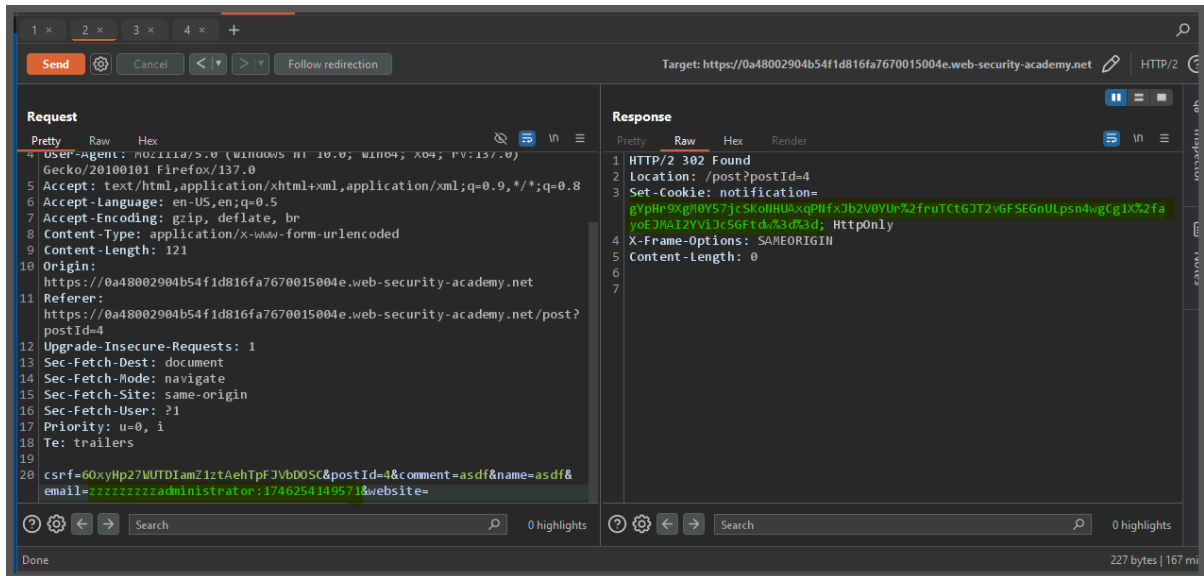
once you got more money
buy leather jacket

Lab: Authentication bypass via encryption oracle

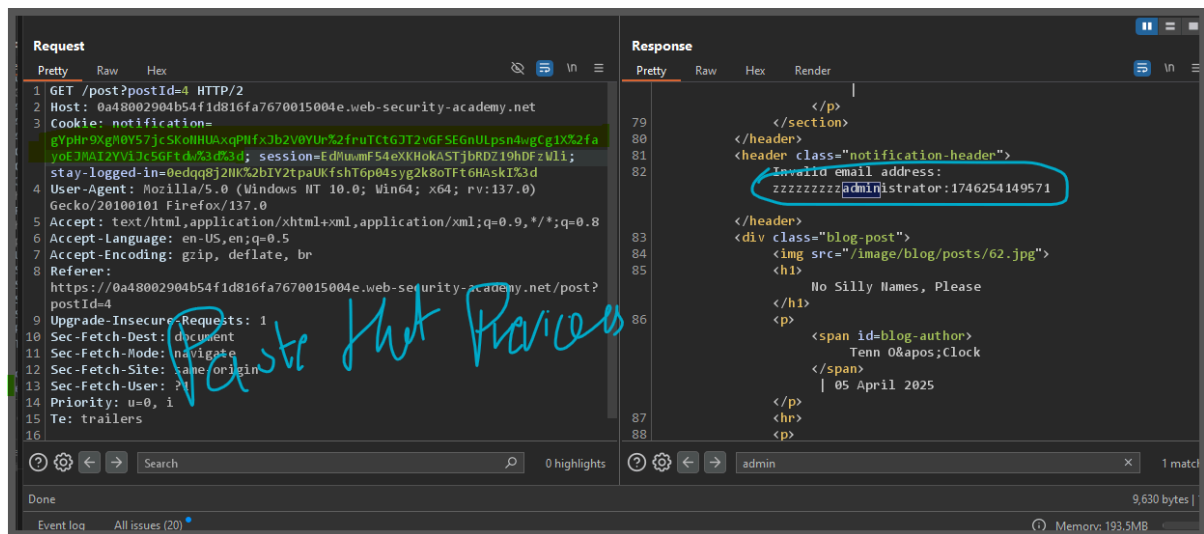
This lab contains a logic flaw that exposes an encryption oracle to users. To solve the lab, exploit this flaw to gain access to the admin panel and delete the user

`carlos` .

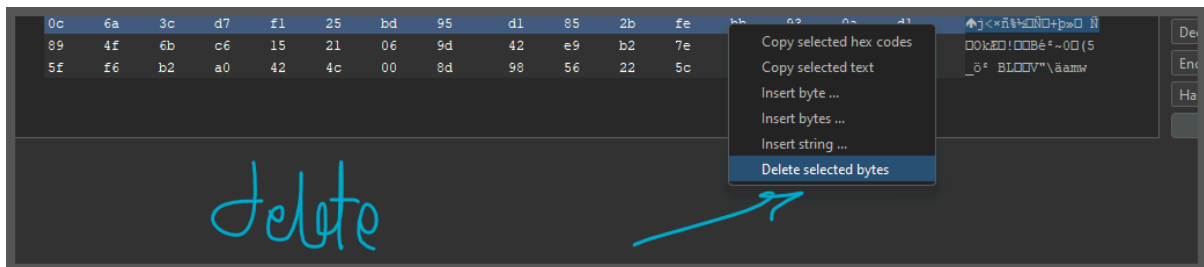
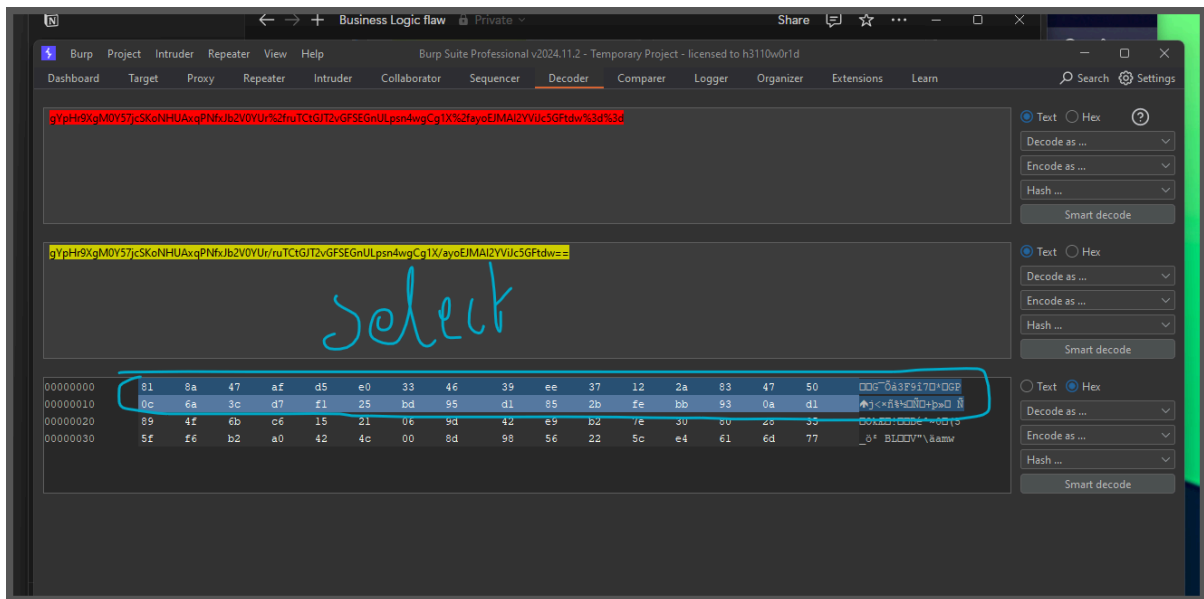
You can log in to your own account using the following credentials: `wiener:peter`



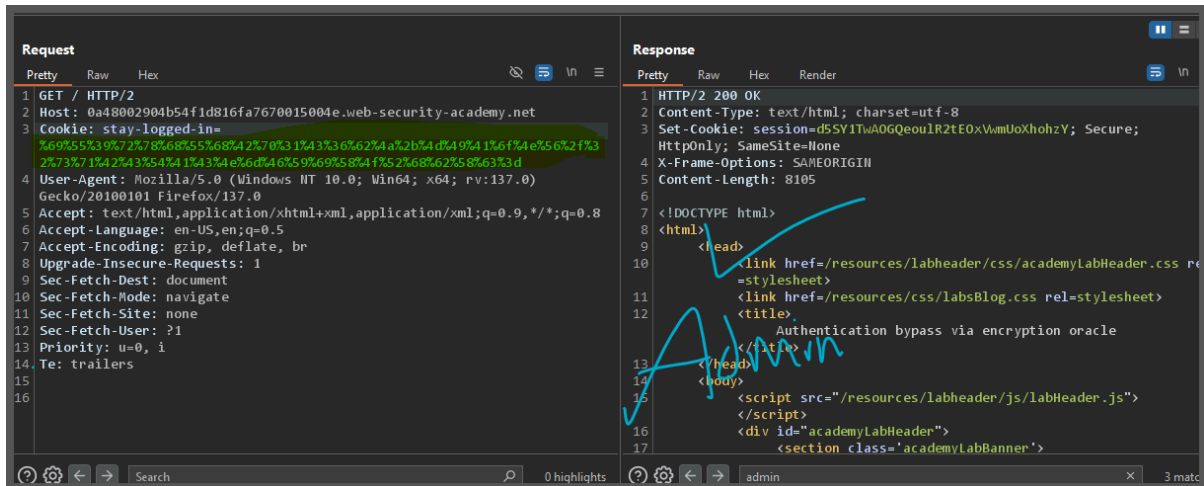
zzzzzzzzzzadministrator:1746254149571



in first screenshot see in response you have got some notification value paste it in second screenshot request



now copy that url encoded and paste it in the
and remove other session cookie only keep stay logged in cookie



Lab: Bypassing access controls using email address parsing discrepancies

This lab validates email addresses to prevent attackers from registering addresses from unauthorized domains. There is a parser discrepancy in the validation logic and library used to parse email addresses.

To solve the lab, exploit this flaw to register an account and delete `carlos`.

solving part is easy

but concept is little bit hard to accept

read this ;<https://portswigger.net/research/splitting-the-email-atom>

The screenshot shows a web application's 'Register' page. The page has a dark theme and includes links for 'Home', 'My account', and 'Register'. The registration form has fields for 'Username', 'Email', and 'Password', and a 'Register' button. A note above the form says: 'If you work for GinAndJuice, please use your @ginandjuice.shop email address'.

```
=?utf-7?q?attacker&AEA-exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.net&ACA-?=@ginandjuice.shop
```

put this in email

and fill other input username and password

you will get an email in fact your email will be considered as owner domain email but inside that is another

Step 1: Identify the format

This string starts with:

```
ruby  
CopyEdit  
=?utf-7?q?
```

That means it's using **MIME encoding**:

- `=?...?...?=' is the format used in email headers (like in "From", "To", or "Subject") to include special characters.`
- `utf-7` means the **character encoding** used is UTF-7.
- `q` means it's using **"quoted-printable" encoding**, often used in email headers.
- So it's a **MIME-encoded email string**.

Step 2: Strip the wrapper

You can extract this part from the MIME wrapper:

```
pgsql  
CopyEdit  
attacker&AEA-exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-serve
```

```
r.net&ACA-
```

Then this part comes after:

```
graphql  
CopyEdit  
=@ginandjuice.shop
```

Now combine them:

```
css  
CopyEdit  
attacker&AEA-exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-serve  
r.net&ACA-@ginandjuice.shop
```

✨ Step 3: Decode special tokens

In **quoted-printable encoding**:

- `&AEA-` = `@`
- `&ACA-` = `.`

So let's **replace** them:

```
css  
CopyEdit  
attacker@exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.ne  
t.@ginandjuice.shop
```

Hmm... this looks weird because there's a **dot just before the @ginandjuice.shop**. It's likely trying to mess with **email validation logic**.

Final Decoded Email (Intended):

Most likely intended to be:

```
CSS
CopyEdit
attacker@exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.net
```

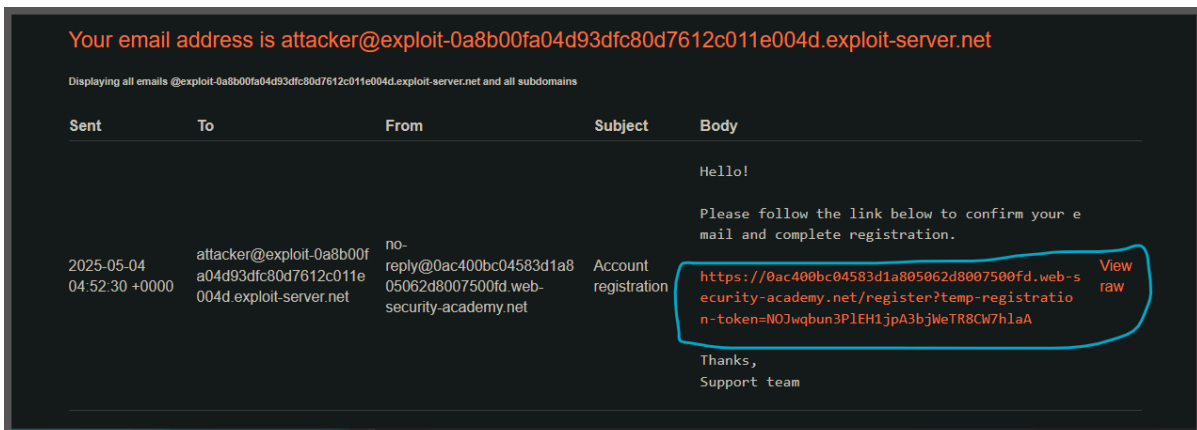
But wrapped in a **MIME-encoded way** and **split across the domain**

@ginandjuice.shop, possibly to bypass filters or exploit bugs in email parsing.

What's the goal of this?

The attacker is:

- **Registering with a weird-looking email address.**
- Trying to **confuse the backend logic** (like how it verifies or validates emails).
- Possibly trying to:
 - Bypass email filters.
 - Hijack flows where emails are auto-sent.
 - Exploit SSRF or misrouted logic in how emails are handled.



Your email address is **attacker@exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.net**

Displaying all emails @exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.net and all subdomains

Sent	To	From	Subject	Body
2025-05-04 04:52:30 +0000	attacker@exploit-0a8b00fa04d93dfc80d7612c011e004d.exploit-server.net	no-reply@0ac400bc04583d1a805062d8007500fd.web-security-academy.net	Account registration	<p>Hello!</p> <p>Please follow the link below to confirm your email and complete registration.</p> <p>https://0ac400bc04583d1a805062d8007500fd.web-security-academy.net/register?temp-registration-token=NOJwqbun3P1EH1jpA3bjWeTR8CW7h1aA</p> <p>Thanks, Support team</p>

[View raw](#)

you your admin

