

Lab 5 - Web Application Firewall (WAF)

Task 1 - Blocking SQLi with WAF

1. Deploy Juice Shop

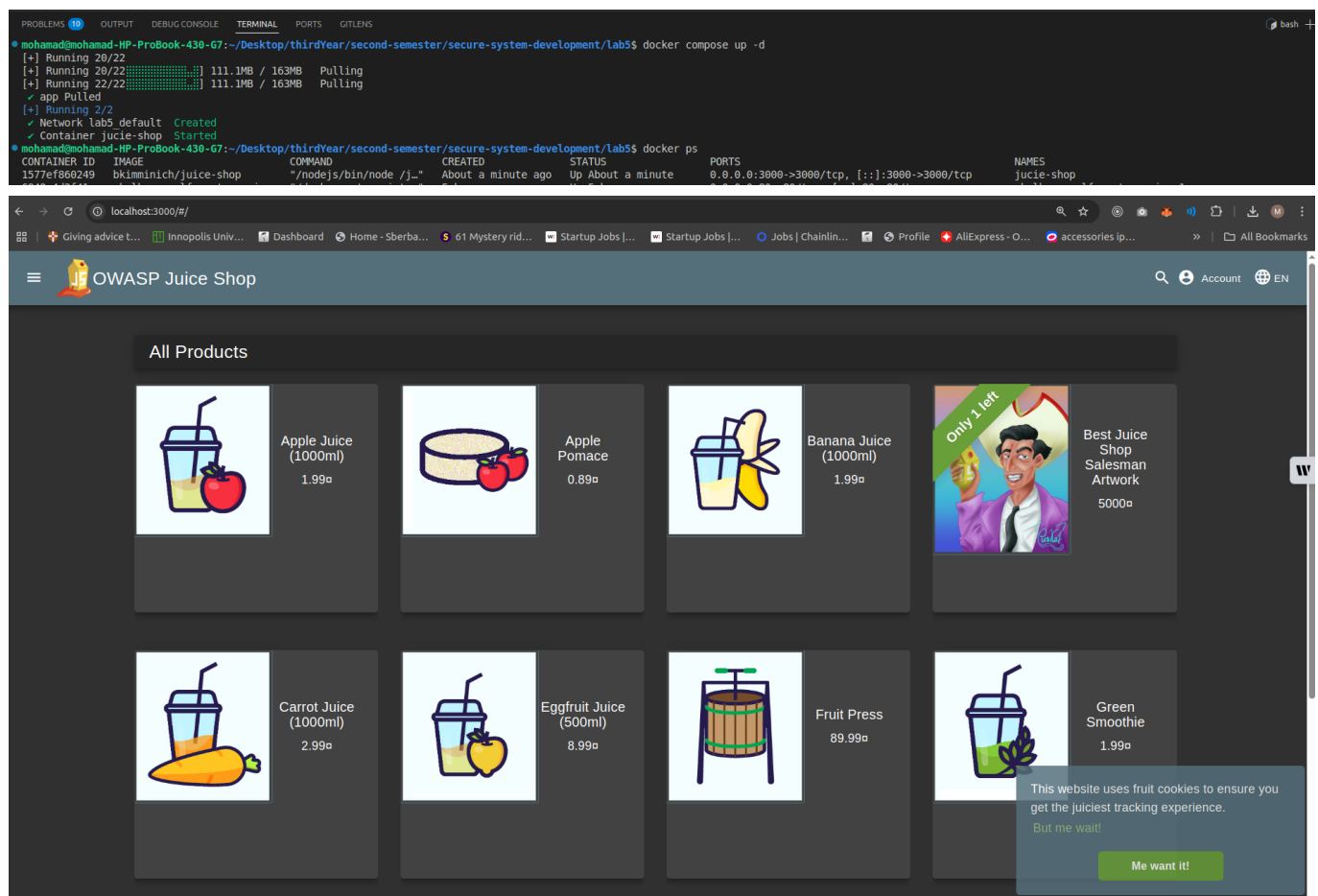
I created a `docker-compose.yaml` file to deploy Juice Shop on port `3000`:

```
app:
  container_name: jucie-shop
  image: bkimminich/juice-shop
  ports:
    - "3000:3000"
```

To launch it:

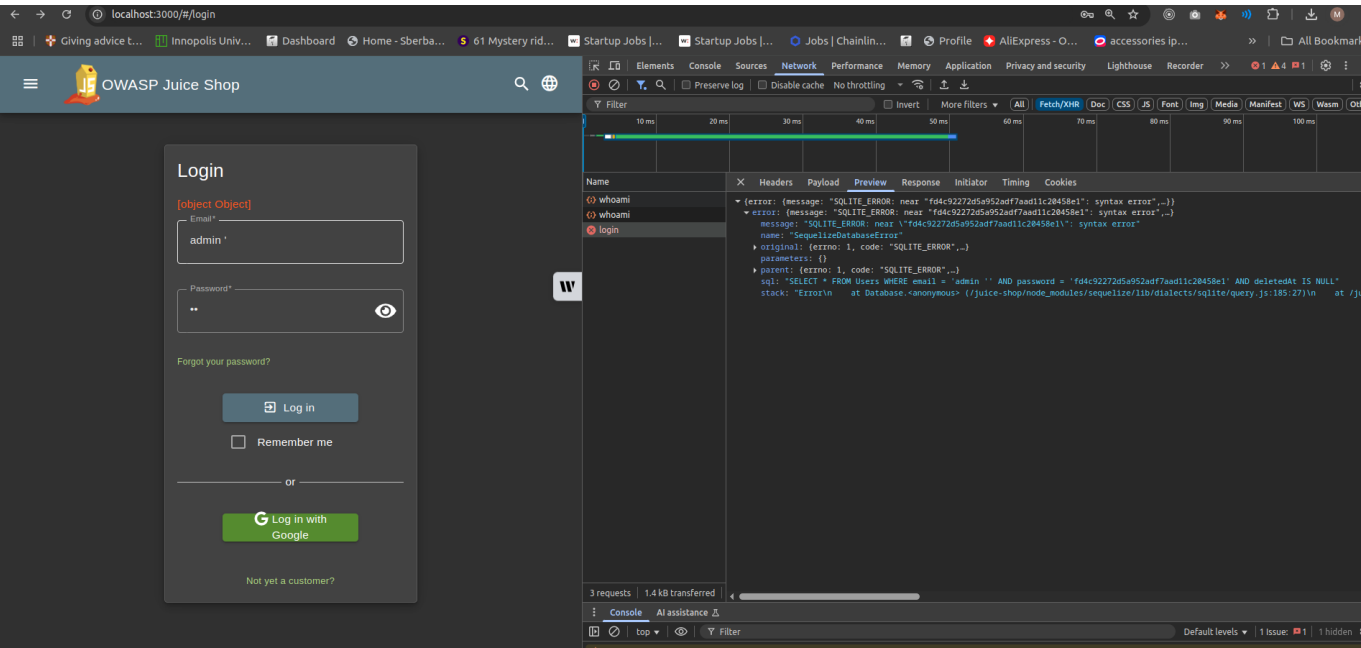
```
docker compose up -d
```

Then accessed the app at <http://localhost:3000>.



2. SQL Injection Exploit

I attempted an SQLi in the login form's **email** field using:

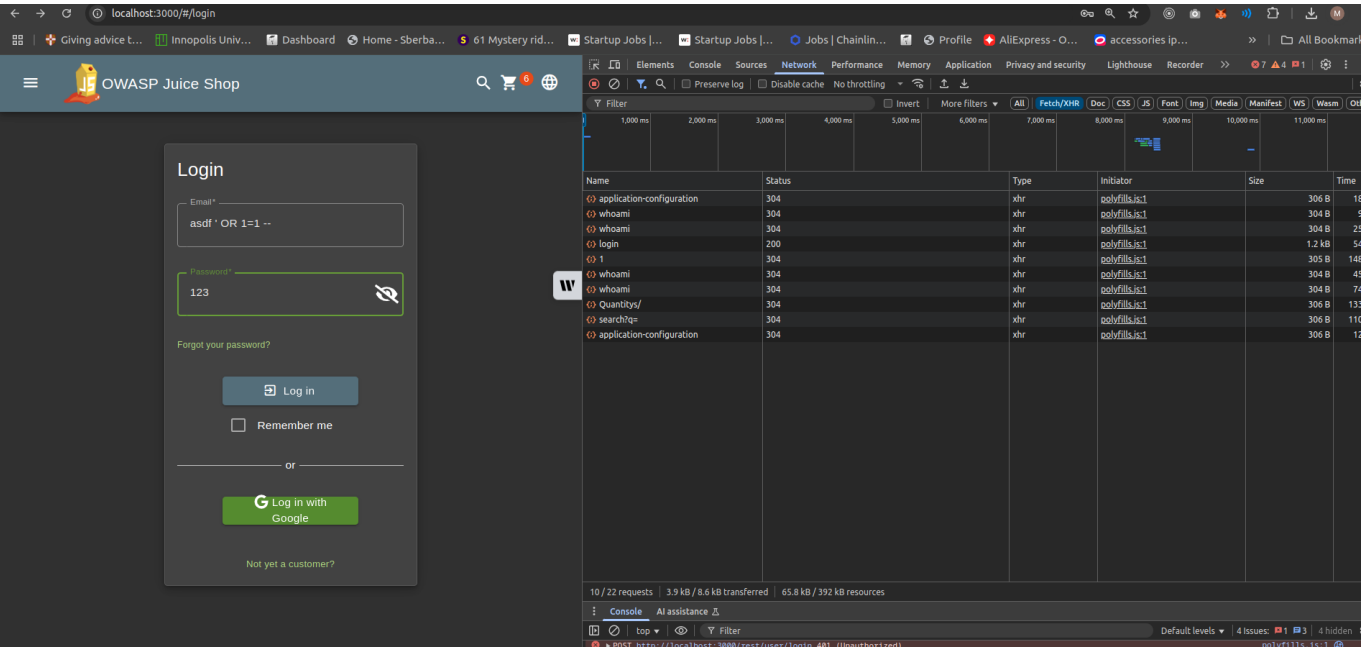


```
asdf ' OR 1=1 --
```

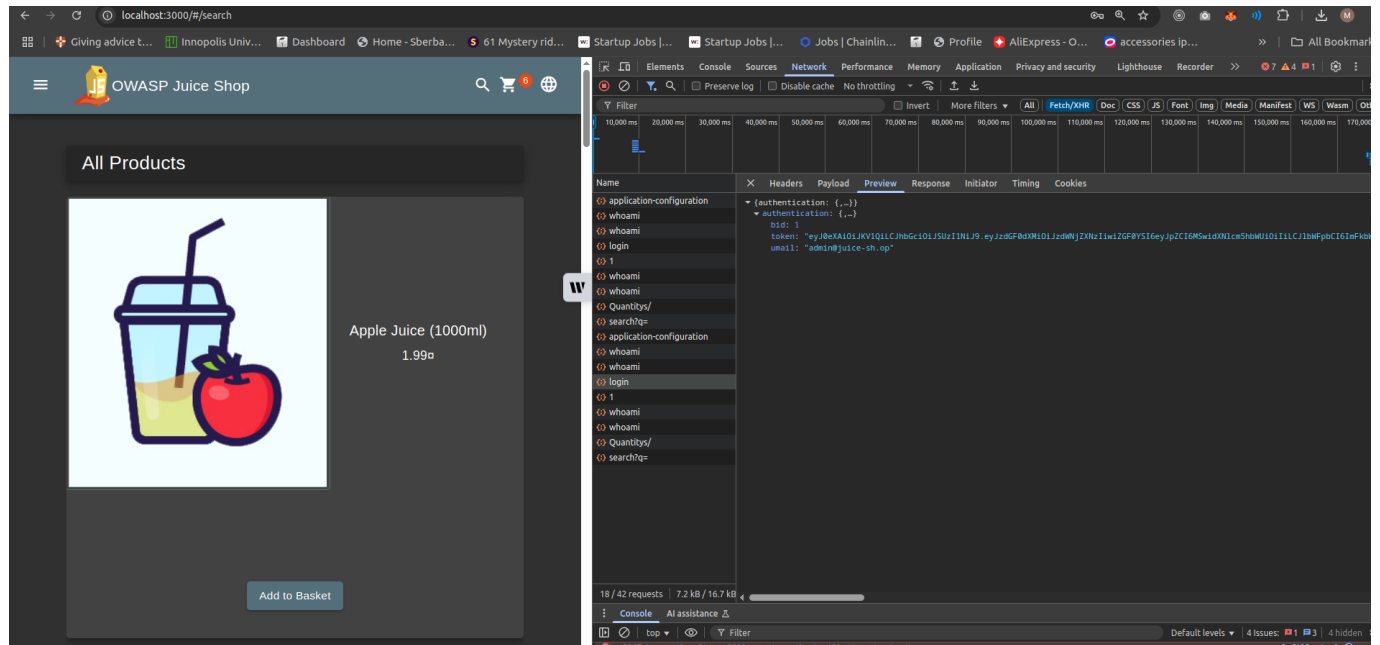
- asdf ' closes the string.
- OR 1=1 always returns true.
- -- comments out the rest.

This allowed me to login as admin.

Before SQLi:



After SQLi:



3. Deploy WAF with ModSecurity + CRS

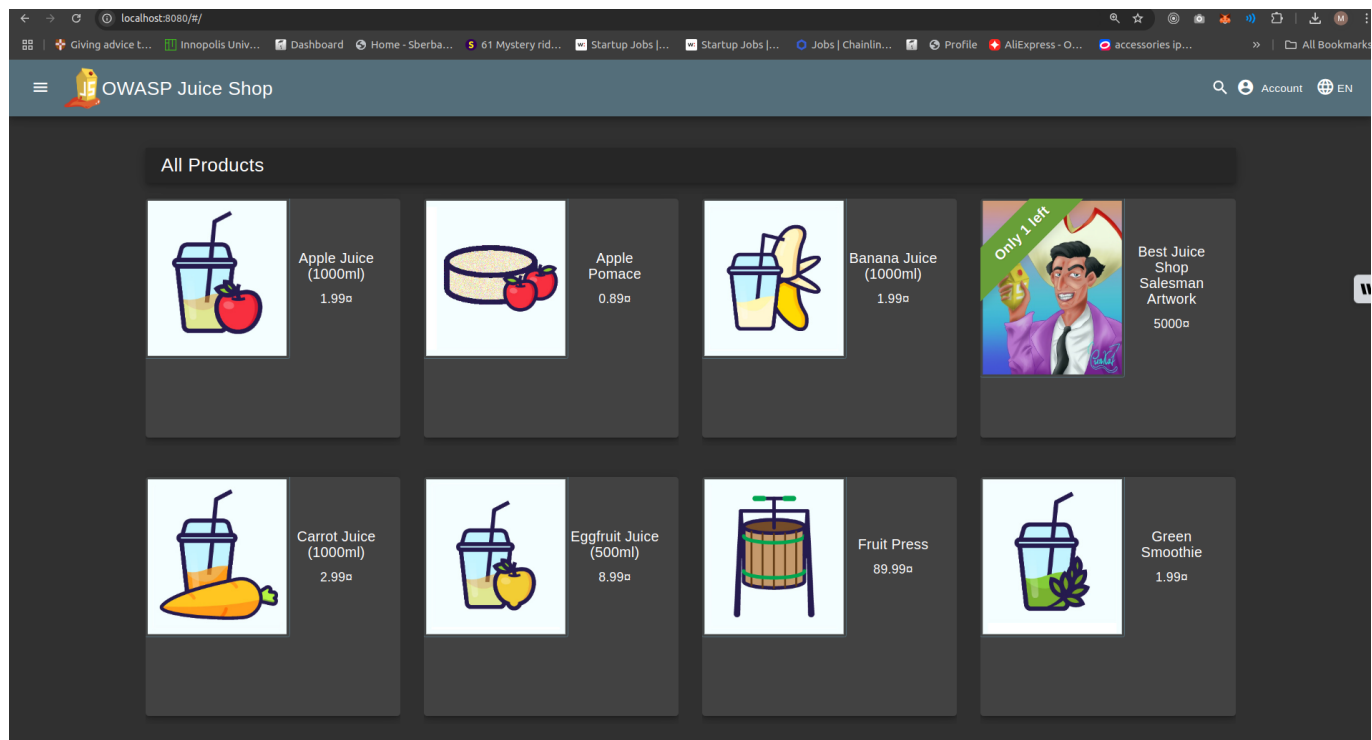
I updated my Docker Compose to add a WAF container running [owasp/modsecurity-crs:nginx], and placed both services on the same network:

```
services:
  app:
    container_name: jucie-shop
    image: bkimminich/juice-shop
    networks:
      - appnet

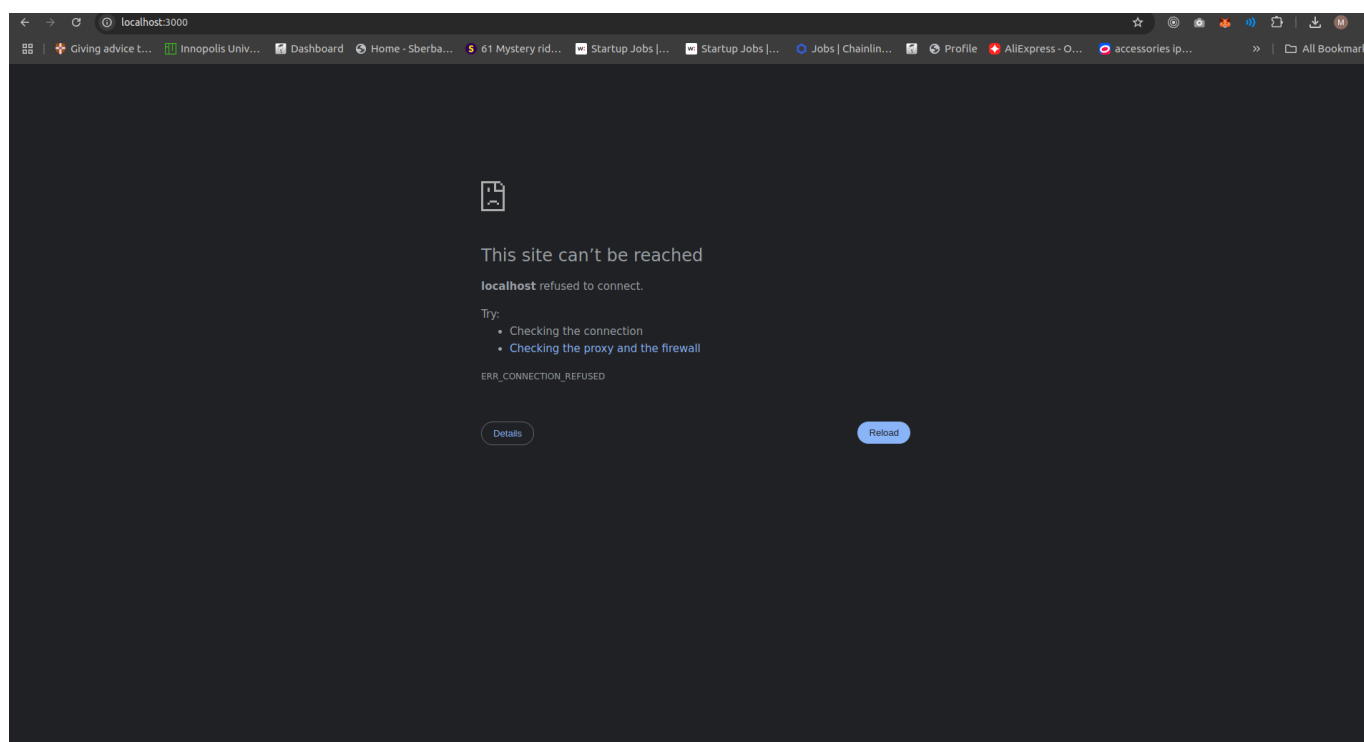
  waf:
    container_name: web-application-firewall
    image: owasp/modsecurity-crs:nginx
    environment:
      - BACKEND=http://app:3000
    ports:
      - "8080:8080"
    networks:
      - appnet

networks:
  appnet:
    driver: bridge
```

Now, Juice Shop is only accessible through the WAF at port 8080:

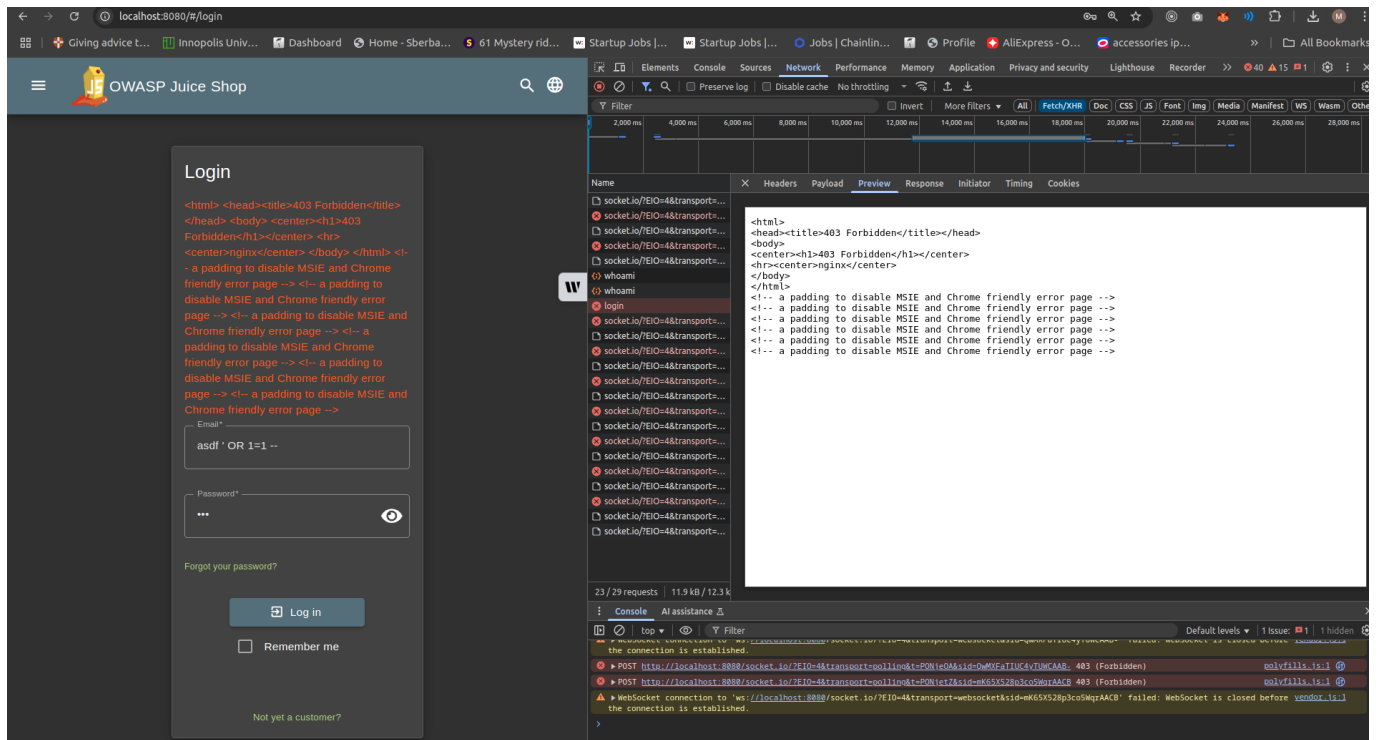


Direct access via port 3000 is disabled:



4. WAF Blocks Original Exploit

The SQLi payload that previously worked (`asdf ' OR 1=1 --`) is now **blocked** by default CRS rules:



Task 2 - Bypassing WAF

1. Update Exploit to Bypass WAF

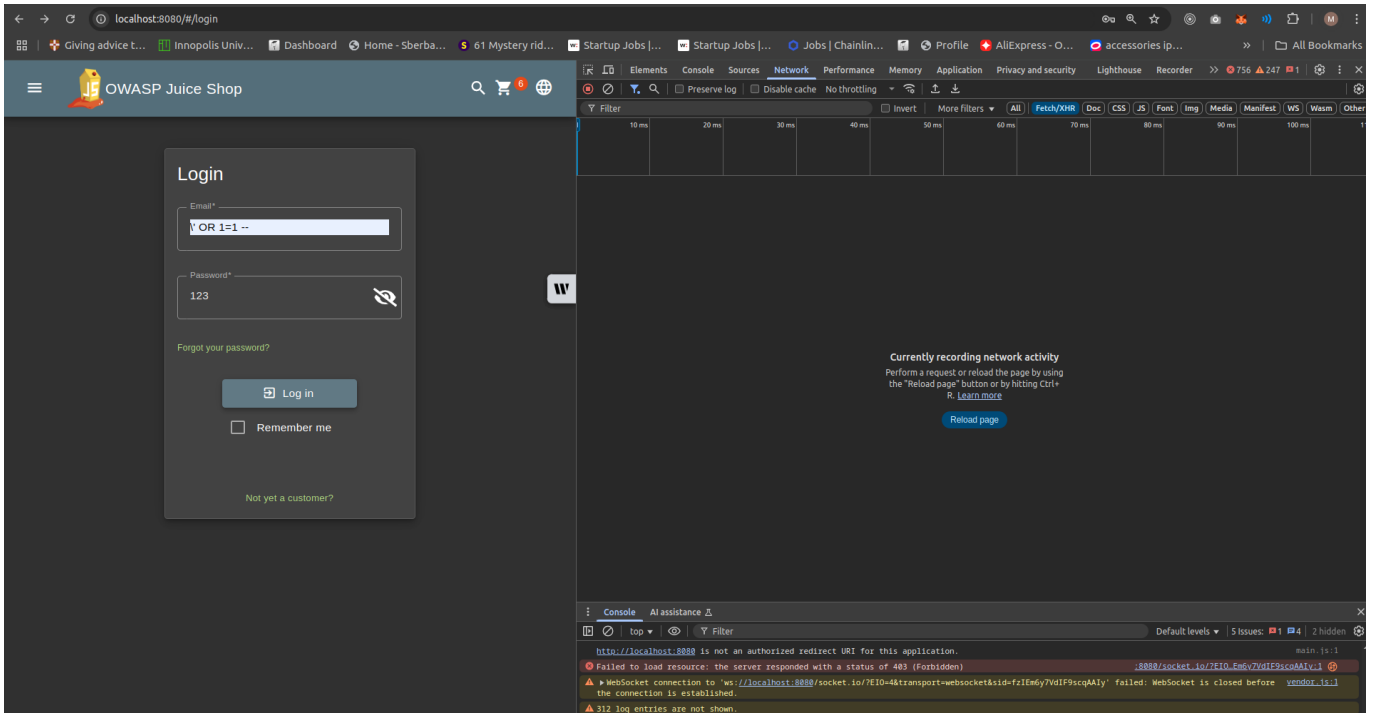
I used a WAF evasion technique from [YesWeHack](#):

```
\' OR 1=1 --
```

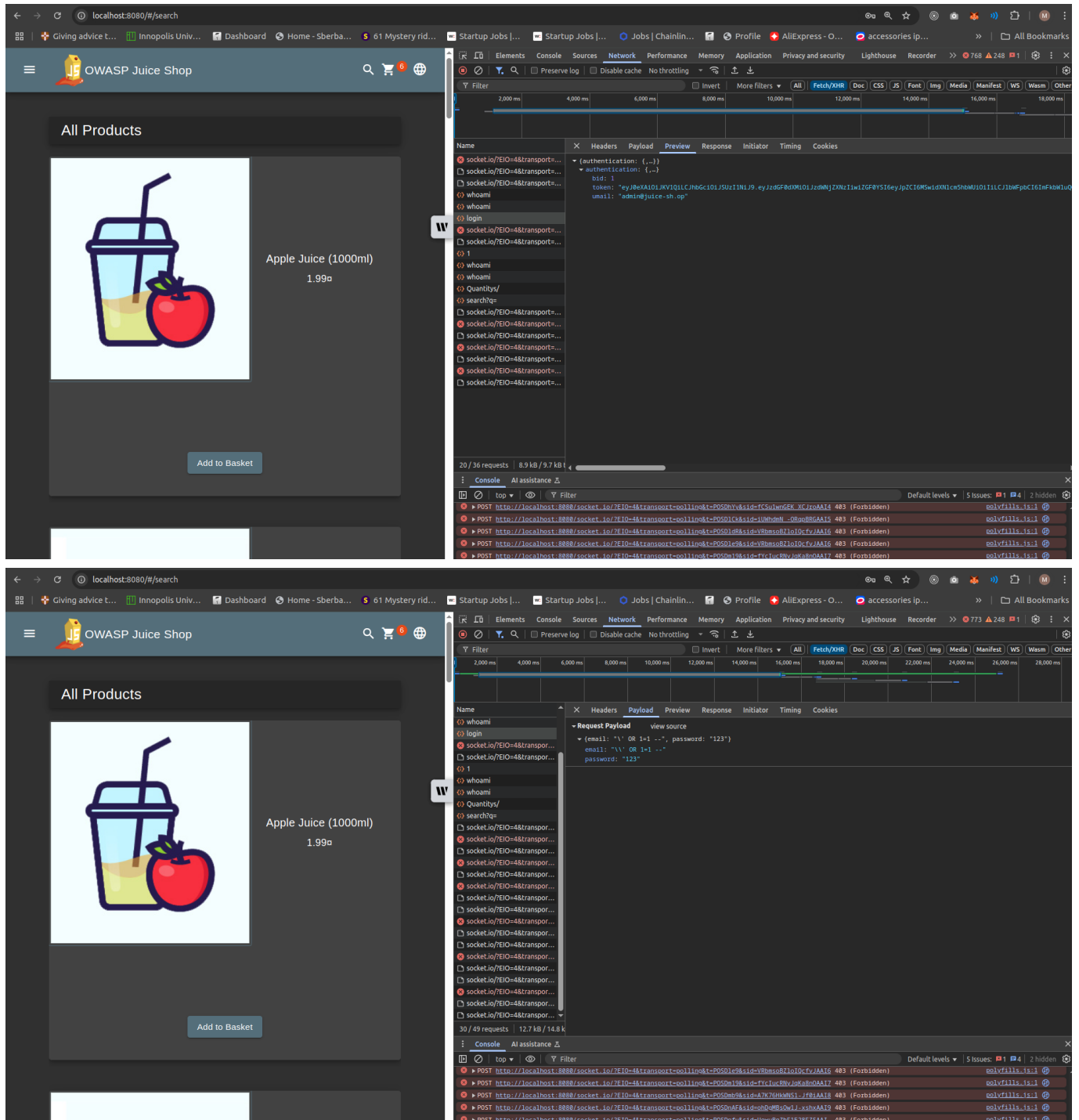
This payload bypassed the default CRS:

- It escapes the quote as `\'`
- Then uses a standard tautology (`OR 1=1`)
- Comments out the rest

Bypass in action:



The screenshot shows a web browser window with the address bar at `localhost:8080/#/login`. The page title is "OWASP Juice Shop". The login form has an "Email*" field containing the text `' OR 1=1 --` and a "Password*" field containing `123`. Below the password field is a "Remember me" checkbox and a "Log in" button. A "Forgot your password?" link is also present. The network console shows a message: "Currently recording network activity. Perform a request or reload the page by using the 'Reload page' button or by hitting Ctrl+R. [Learn more](#)". The console also displays several error messages: "http://localhost:8080 is not an authorized redirect URI for this application.", "Failed to load resource: the server responded with a status of 403 (Forbidden)", and "WebSocket connection to 'ws://localhost:8080/socket.io/?EIO=4&transport=websocket&sid=fz1Emdy7NdF9scqAAlY' failed: WebSocket is closed before the connection is established." The console also shows a warning: "312 log entries are not shown".



2. Add Custom Rule to Block Bypassed Exploit

I created a file `custom-rule-sqli.conf` with the following rule:

```
SecRule ARGS|ARGS_NAMES|REQUEST_URI|REQUEST_HEADERS|REQUEST_BODY "@rx (?i:
(\\'|"%5C%27)\\s*
(or|and|union|select|insert|group|having|benchmark|sleep)\\b)" \
    "id:100102,phase:2,t:none,t:urlDecodeUni,log,deny,status:403,msg:'SQLi:
escaped quote followed by SQL keyword',severity:CRITICAL"
```

 Explanation

- **Targets:** All request fields (ARGS, headers, URI, body)
- **Regex:** Matches `\'` or `%5C%27` followed by common SQL keywords
- **t:urlDecodeUni:** Decodes URL encoding before matching
- **deny,status:403:** Blocks the request

Mounted the rule in `docker-compose.yml`:

```
volumes:
  - ./custom-rule-sqli.conf:/etc/modsecurity.d/owasp-crs/rules/custom-rule-sqli.conf:ro
```

Confirmed it's loaded:

```
$ ls /etc/modsecurity.d/owasp-crs/rules/
REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example  REQUEST-944-APPLICATION-ATTACK-JAVA.conf      php-config-directives.data
REQUEST-901-INITIALIZATION.conf                      REQUEST-949-BLOCKING-EVALUATION.conf          php-errors-pl2.data
REQUEST-905-COMMON-EXCEPTIONS.conf                   RESPONSE-950-DATA-LEAKAGES.conf              php-errors.data
REQUEST-911-METHOD-ENFORCEMENT.conf                 RESPONSE-951-DATA-LEAKAGES-SQL.conf           php-function-names-933150.data
REQUEST-913-SCANNER-DETECTION.conf                   RESPONSE-952-DATA-LEAKAGES-JAVA.conf          php-function-names-933151.data
REQUEST-920-PROTOCOL-ENFORCEMENT.conf                 RESPONSE-953-DATA-LEAKAGES-PHP.conf           php-variables.data
REQUEST-921-PROTOCOL-ATTACK.conf                     RESPONSE-954-DATA-LEAKAGES-IIS.conf           restricted-files.data
REQUEST-922-MULTIPART-ATTACK.conf                     RESPONSE-955-WEB-SHELLS.conf                 restricted-upload.data
REQUEST-930-APPLICATION-ATTACK-LFI.conf               RESPONSE-959-BLOCKING-EVALUATION.conf         scanners-user-agents.data
REQUEST-931-APPLICATION-ATTACK-RFI.conf               RESPONSE-980-CORRELATION.conf                 sql-errors.data
REQUEST-932-APPLICATION-ATTACK-RCE.conf               RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf.example  ssrf.data
REQUEST-933-APPLICATION-ATTACK-PHP.conf               custom-rule-sqli.conf                        unix-shell.data
REQUEST-934-APPLICATION-ATTACK-GENERIC.conf           iis-errors.data                             web-shells-php.data
REQUEST-941-APPLICATION-ATTACK-XSS.conf               java-classes.data                          windows-powershell-commands.data
REQUEST-942-APPLICATION-ATTACK-SQLI.conf              java-errors.data
REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION.conf  lfi-os-files.data
```

3. Apply Config & Retest

After restarting the WAF, I tested again:

- Input: `\' OR 1=1 --`
- Result: Blocked


```

mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/secure-system-development/Lab5$ docker compose up -d
[+] Running 1/1
  ✓ waf Pulled
[+] Running 3/3
  ✓ Network lab5_apynet Created
  ✓ Container jucie-shop Started
  ✓ Container web-application-firewall Started
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/secure-system-development/Lab5$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
05a22639e88a   owasp/modsecurity-crs:nginx         "/docker-entrypoint..." 7 seconds ago  Up 7 seconds  0.0.0.0:8080->8080/tcp   web-application-firewall
e0819ab3da8d   bkimminich/juice-shop              "/nodejs/bin/node /j..." 7 seconds ago  Up 7 seconds  3000/tcp                 jucie-shop
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/secure-system-development/Lab5$ docker exec -it 05a22639e88a sh
$ ls
bin boot dev docker-entrypoint.d docker-entrypoint.sh etc home lib lib64 media mnt opt proc root run/sbin srv sys tmp usr var
$ cat /etc/modsecurity.d/owasp-crs/rules/custom-rule-sqli.conf
SecRule ARGS|ARGS_NAMES|REQUEST_URI|REQUEST_HEADERS|REQUEST_BODY "@rx (?i:\\\\'%5C%27)\\s*(or|and|union|select|insert|group|having|benchmark|sleep)\\b)" \
  "id:100102,phase:2,t:none,t:urlDecodeUni,log,deny,status:403,msg:'SQLi: escaped quote followed by SQL keyword',severity:CRITICAL"
$ cat /etc/modsecurity.d/setup.conf
# Note: the plugin rules will be uncommented when the container starts,
# depending on whether the respective files exist. This works around
# the issue that ModSecurity doesn't support optional includes on NGINX.

# Allow custom rules to be specified in:
# /opt/modsecurity/rules/{before,after}-crs/*.conf

Include /etc/modsecurity.d/modsecurity.conf
Include /etc/modsecurity.d/modsecurity-override.conf

Include /etc/modsecurity.d/owasp-crs/crs-setup.conf

Include /etc/modsecurity.d/owasp-crs/plugins/*-config.conf
Include /etc/modsecurity.d/owasp-crs/plugins/*-before.conf

Include /etc/modsecurity.d/owasp-crs/rules/*.conf

Include /etc/modsecurity.d/owasp-crs/plugins/*-after.conf
$

```

The screenshot shows a web browser at localhost:8080/#/login. The page displays a "Login" form with fields for "Email" and "Password". The "Email" field contains the text "\' OR 1=1 --". The "Password" field contains "123". Below the form, there is a "Log in" button and a "Remember me" checkbox. A "403 Forbidden" error message is visible at the top of the page. The browser's developer tools are open, showing the "Network" tab with a list of requests. The "login" request is highlighted, showing a failed status (403) and a response body containing the error message. The "Console" tab shows a message about a Websocket connection failure.

4. Automated SQLi Test with sqlmap

Ran this command:

```

python3 sqlmap.py -u "http://localhost:8080/rest/user/login" \
  --data='{"email":"\\\' OR 1=1 --", "password":"x"}' \
  --headers="Content-Type: application/json" \
  --level=5 --risk=3 \
  --technique=BEUSTQ \
  --random-agent --batch -v 3 \
  --ignore-code 401

```

Before Custom Rule:

403 (Forbidden) - 28845 times

After Custom Rule:

403 (Forbidden) - 41406 times

```
[20:10:53] [PAYLOAD] -9882')))) ORDER BY 1#
[20:10:53] [PAYLOAD] -9427' ORDER BY 1#
[20:10:53] [PAYLOAD] -5608') ORDER BY 1#
[20:10:53] [PAYLOAD] -1705')) ORDER BY 1#
[20:10:53] [PAYLOAD] -3071')) ORDER BY 1#
[20:10:53] [PAYLOAD] -8308' ORDER BY 1#
[20:10:53] [PAYLOAD] -1270' ORDER BY 1#
[20:10:53] [PAYLOAD] -6608' ORDER BY 1#
[20:10:53] [PAYLOAD] -8596')) ORDER BY 1#
[20:10:53] [PAYLOAD] -1674')) ORDER BY 1#
[20:10:53] [PAYLOAD] -6608' ORDER BY 1#
[20:10:53] [PAYLOAD] -4357' ORDER BY 1#
[20:10:53] [PAYLOAD] -3253')) ORDER BY 1#
[20:10:53] [PAYLOAD] -8688')) ORDER BY 1#
[20:10:53] [PAYLOAD] -9067' ORDER BY 1#
[20:10:53] [PAYLOAD] -8555 ORDER BY 1#
[20:10:53] [PAYLOAD] -4910 ORDER BY 1#
[20:10:53] [PAYLOAD] -5903' ORDER BY 1#
[20:10:53] [PAYLOAD] -3274')) AS Svdy WHERE 4602=4602 ORDER BY 1#
[20:10:53] [PAYLOAD] -3428')) AS swm WHERE 4894=4894 ORDER BY 1#
[20:10:53] [PAYLOAD] -1597)) AS ndbe WHERE 4460=4460 ORDER BY 1#
[20:10:53] [PAYLOAD] -1895')) AS nUll WHERE 4131=4131 ORDER BY 1#
[20:10:53] [PAYLOAD] -4055')) AS fNv WHERE 7200=7200 ORDER BY 1#
[20:10:53] [PAYLOAD] -9644) AS kp03 WHERE 8202=8202 ORDER BY 1#
[20:10:53] [PAYLOAD] -6638' WHERE 9405=9405 ORDER BY 1#
[20:10:53] [PAYLOAD] -5428') WHERE 4735=4735 ORDER BY 1#
[20:10:53] [PAYLOAD] -1007' IN BOOLEAN MODE) ORDER BY 1#
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (NULL) - 11 to 20 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (random number) - 11 to 20 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (NULL) - 21 to 30 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (random number) - 21 to 30 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (NULL) - 31 to 40 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (random number) - 31 to 40 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (NULL) - 41 to 50 columns'
[20:10:53] [DEBUG] skipping test 'MySQL UNION query (random number) - 41 to 50 columns'
[20:10:53] [WARNING] parameter 'Host' does not seem to be injectable
[20:10:53] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper'
--tamper=space2comment')
[20:10:53] [WARNING] HTTP error codes detected during run:
401 (Unauthorized) - 8 times, 403 (Forbidden) - 28845 times
[20:10:53] [DEBUG] too many 4xx and/or 5xx HTTP error codes could mean that some kind of protection is involved (e.g. WAF)
[*] ending @ 20:10:53 /2025-04-11/
```

```
[20:23:43] [PAYLOAD] -1805)) AS uPwd WHERE 6533=6533 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -5069)) AS lpat WHERE 3055=3055 ORDER BY 9024#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -1557')) AS ZqLL WHERE 9141=9141 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -1315') AS uWns WHERE 3627=3627 ORDER BY 9381#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -3092')) AS PuG0 WHERE 8116=8116 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -2294')) AS vBba WHERE 7484=7484 ORDER BY 4854#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -5201) AS wabD WHERE 2908=2908 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -9369) AS hLw WHERE 1030=1030 ORDER BY 6093#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -2329' WHERE 3188=3188 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -5632' WHERE 7505=7505 ORDER BY 5372#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -6832') WHERE 4167=4167 ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -2705')) WHERE 5903=5903 ORDER BY 3952#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -5229' IN BOOLEAN MODE) ORDER BY 1#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [PAYLOAD] -2856' IN BOOLEAN MODE) ORDER BY 9101#
[20:23:43] [DEBUG] got HTTP error code: 403 ('Forbidden')
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (NULL) - 11 to 20 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (random number) - 11 to 20 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (NULL) - 21 to 30 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (random number) - 21 to 30 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (NULL) - 31 to 40 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (random number) - 31 to 40 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (NULL) - 41 to 50 columns'
[20:23:43] [DEBUG] skipping test 'MySQL UNION query (random number) - 41 to 50 columns'
[20:23:43] [WARNING] parameter 'Host' does not seem to be injectable
[20:23:43] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper'
--tamper=space2comment')
[20:23:43] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 41406 times, 502 (Bad Gateway) - 1 times, 401 (Unauthorized) - 7 times
[20:23:43] [DEBUG] too many 4xx and/or 5xx HTTP error codes could mean that some kind of protection is involved (e.g. WAF)
[*] ending @ 20:23:43 /2025-04-11/
```

Final Results Summary

Before adding the custom rule, the WAF blocked about 29,000 SQL injection attempts using default CRS rules. After adding my custom rule, over 41,000 requests were blocked. This shows that the custom rule improved security by catching more SQL injection patterns that might bypass default protections.

[Github Link](#)
