

Attack Vectors:

1. SQL Injection Attack
2. File size restriction
3. Firewall rule to blacklist IP.

1. SQL Injection Attack:

Below is the SQL injection attack performed on the WebApp.

Command:

```
python sqlmap.py -u "https://csye6225-spring2019-murgoda.me/user/register/" -
-method=POST --
data={"email":"qwertyqwerty@gmail.com","password":"Test@123"}' --
tamper=space2comment
```

SQL injection attack results:

a. WAF Enabled:

When Web Application Firewall is enabled SQL injection attack is forbidden with 403 error code as shown below.

```

[14:45:11] [INFO] testing if (custom) POST parameter 'JSON email' is dynamic
[14:45:11] [INFO] (custom) POST parameter 'JSON email' appears to be dynamic
[14:45:11] [INFO] testing for SQL injection on (custom) POST parameter 'JSON email'
[14:45:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:45:12] [INFO] testing 'boolean-based blind - Parameter replace (original value)'
[14:45:12] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:45:13] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:45:13] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:45:13] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:45:14] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[14:45:14] [INFO] testing 'MySQL inline queries'
[14:45:14] [INFO] testing 'PostgreSQL inline queries'
[14:45:14] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:45:14] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:45:14] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:45:15] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:45:15] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[14:45:15] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind (IF)'
[14:45:15] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[14:45:16] [INFO] testing 'Oracle AND time-based blind'
[14:45:16] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:45:16] [WARNING] (custom) POST parameter 'JSON email' does not seem to be injectable
[14:45:16] [INFO] testing if (custom) POST parameter 'JSON password' is dynamic
[14:45:16] [INFO] (custom) POST parameter 'JSON password' appears to be dynamic
[14:45:16] [INFO] testing for SQL injection on (custom) POST parameter 'JSON password'
[14:45:16] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:45:16] [INFO] testing 'boolean-based blind - Parameter replace (original value)'
[14:45:16] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:45:16] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:45:16] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:45:16] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:45:16] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[14:45:16] [INFO] testing 'MySQL inline queries'
[14:45:16] [INFO] testing 'PostgreSQL inline queries'
[14:45:16] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:45:16] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:45:16] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:45:16] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:45:16] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[14:45:16] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:45:16] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[14:45:16] [INFO] testing 'Oracle AND time-based blind'
[14:45:16] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:45:16] [WARNING] (custom) POST parameter 'JSON password' does not seem to be injectable
[14:45:16] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests
[14:45:16] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 92 times, 502 (Bad Gateway) - 55 times

[*] ending @ 14:45:19 / 2019-04-04/

```

b. WAF disabled:

When Web Application Firewall is disabled, firewall doesn't forbid the attack, however the API call is denied with error 502 (Bad Gateway) as the web application framework provides the security against SQL injection attack.

```

akshay@ubuntu: ~/cloud6225/Assignment8/sqlmap
[*] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 18:21:57 /2019-04-04/
[18:21:57] [INFO] loading tamper module 'space2comment'
JSON data found in POST data. Do you want to process it? [Y/n/q] Y
[18:22:00] [INFO] testing connection to the target URL
[18:22:00] [INFO] heuristics detected web page charset 'ascii'
[18:22:00] [WARNING] the web server responded with an HTTP error code (502) which could interfere with the results of the tests
[18:22:00] [INFO] testing if the target URL content is stable
[18:22:00] [INFO] target URL content is stable
[18:22:00] [INFO] testing if (custom) POST parameter 'JSON_email' is dynamic
[18:22:00] [INFO] (custom) POST parameter 'JSON_email' appears to be dynamic
[18:22:00] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON_email' might not be injectable
[18:22:00] [INFO] testing for SQL injection on (custom) POST parameter 'JSON_email'
[18:22:00] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:22:00] [INFO] (custom) POST parameter 'JSON_email' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --code=200)
[18:22:00] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'H2'
It looks like the back-end DBMS is 'H2'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'H2' extending provided level (1) and risk (1) values? [Y/n] Y
[18:22:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[18:22:00] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[18:22:00] [INFO] checking if the injection point on (custom) POST parameter 'JSON_email' is a false positive
[18:22:00] [WARNING] false positive or unexploitable injection point detected
[18:22:00] [WARNING] (custom) POST parameter 'JSON_email' does not seem to be injectable
[18:22:00] [INFO] testing if (custom) POST parameter 'JSON_password' is dynamic
[18:22:00] [INFO] (custom) POST parameter 'JSON_password' does not appear to be dynamic
[18:22:00] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON_password' might not be injectable
[18:22:00] [INFO] testing for SQL injection on (custom) POST parameter 'JSON_password'
[18:22:00] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:22:00] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[18:22:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[18:22:00] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[18:22:00] [INFO] target URL appears to have 18 columns in query
Injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] Y
[18:22:00] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[18:22:00] [WARNING] (custom) POST parameter 'JSON_password' does not seem to be injectable
[18:22:00] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level/'--risk' options if you wish to perform more tests
[18:22:00] [WARNING] HTTP error codes detected during run:
502 (Bad Gateway) - 89 times
[*] ending @ 18:22:31 /2019-04-04/
akshay@ubuntu: ~/cloud6225/Assignment8/sqlmap$ python sqlmap.py -u "https://cysye6225-spring2019-gadhyah.me/user/register/" --method=POST --data="{\"email\":\"qasdfqzxc12@gmail.com\", \"password\":\"Test0123\"}" --tamper=space2comment

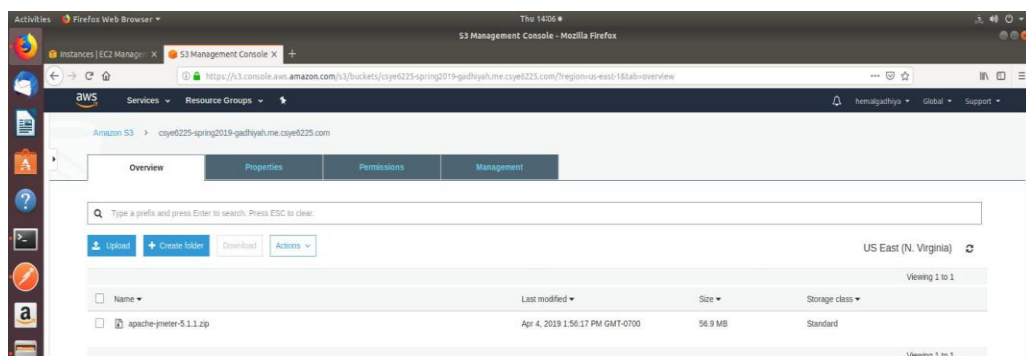
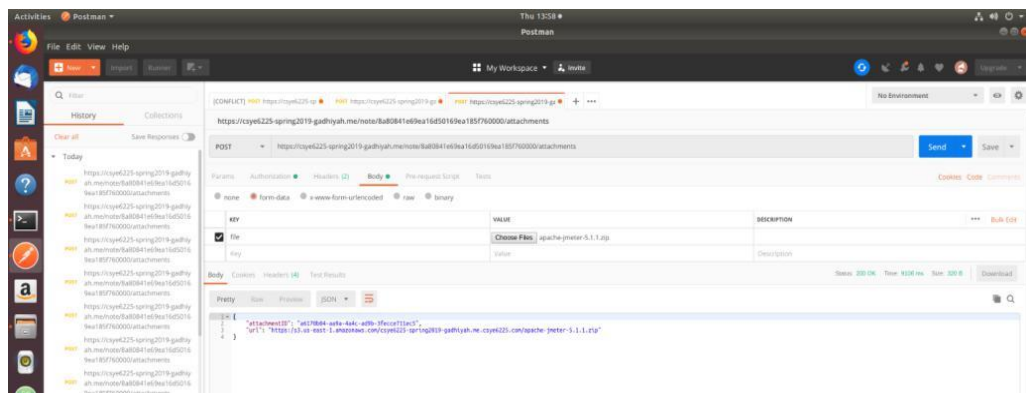
```

2. File size restriction:

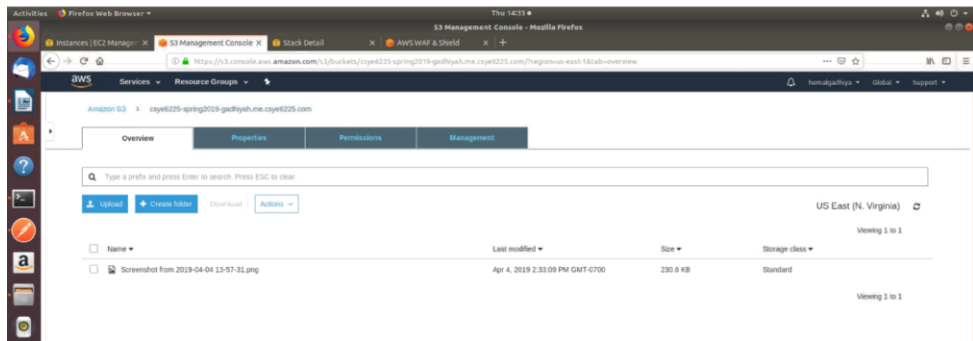
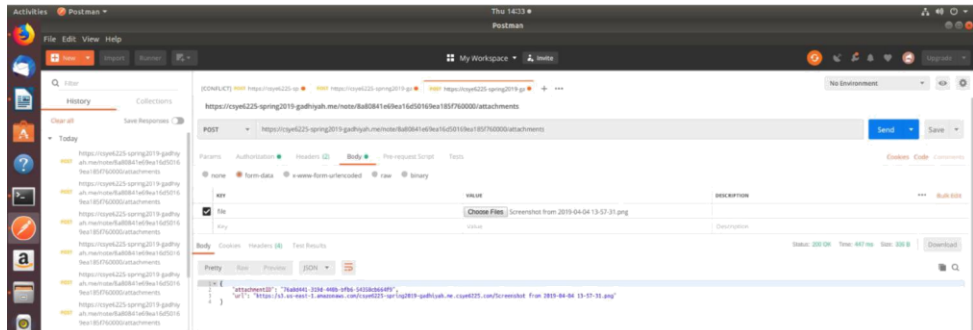
Firewall rule is added to block the file attachment to note if file size greater than 1MB.

Without firewall: File with size 56 MB successfully attached to note and stored in S3 as there is no such restriction.

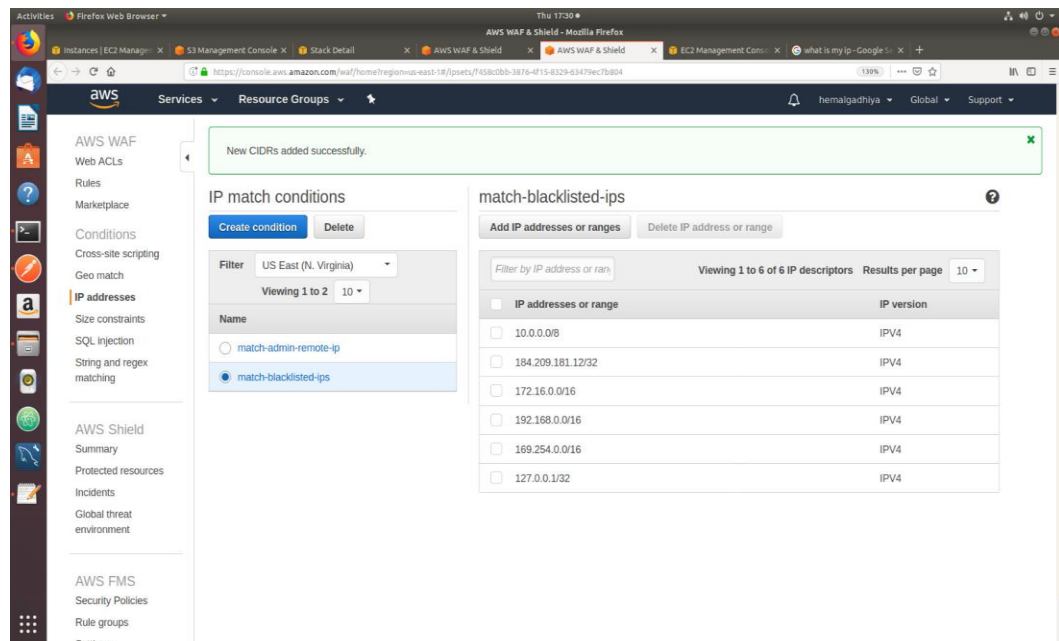
- WAF disabled (To restrict file size):
Successfully able to attach the large sized file and stored into S3



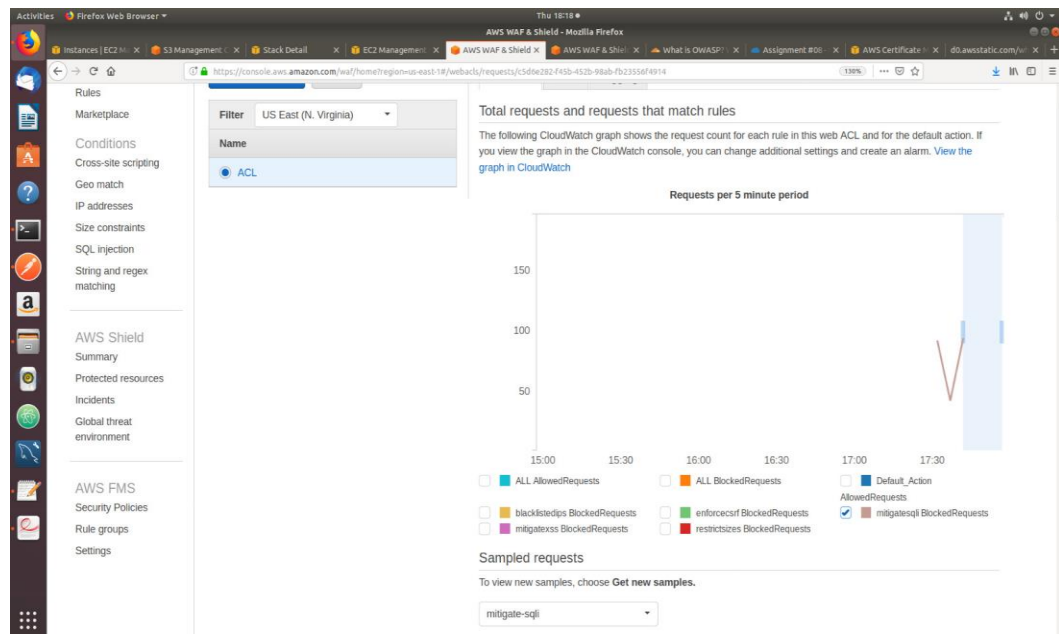
-
- The screenshot shows the Postman application interface. The top bar indicates the user is logged in as 'Postman'. The main workspace is titled 'My Workspace' and 'None'. The URL bar shows the endpoint: `https://cyeet225-spring2019-gadflyh.me/hibo/Bd0841ed5ea16d50165ea165f760000.attachments`. The method is set to **POST**. The body is set to **raw** with a JSON payload: `{ "name": "Formdata", "url": "www.formdata.co.uk", "raw": "binary" }`. The **Headers** tab is active, showing a **Content-Type** header set to `application/json; charset=utf-8`. The **Pre-request Scripts** and **Tests** tabs are also visible. The bottom status bar shows the request is **Ready** and the response is **None**.



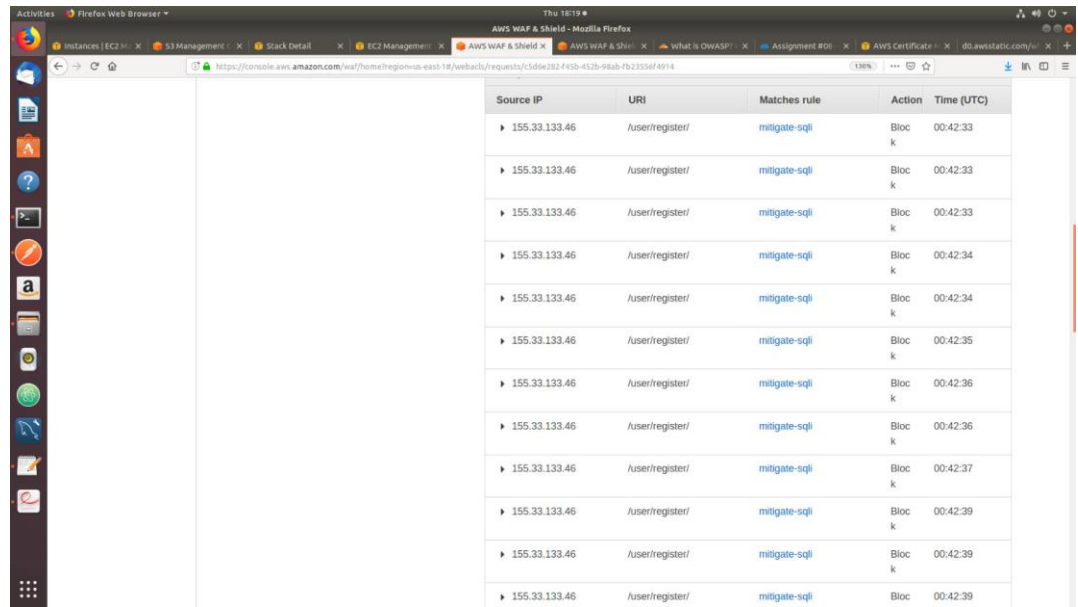
- In our example:
We determine the public IP address of our machine add that IP address to list of blacklisted IP addresses as below:



When API call is made from blacklisted IP's these calls will be blocked by WAF which can be seen from AWS console refer below screenshot:



WEB ATTACK VECTORS



The screenshot shows the AWS WAF console in a Firefox browser. The left sidebar contains navigation links for Instances, EC2, S3 Management, Stack Detail, EC2 Management, AWS WAF & Shield, AWS WAF & Shield, What is OWASP, Assignment #01, AWS Certificate, and awsstatic.com. The main content area displays a table of blocked requests. The table has five columns: Source IP, URI, Matches rule, Action, and Time (UTC). All requests are from the source IP 155.33.133.46 to the URI /user/register/. They all match the rule mitigate-sqli and are blocked. The times range from 00:42:33 to 00:42:39 UTC.

Source IP	URI	Matches rule	Action	Time (UTC)
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:33
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:33
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:33
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:34
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:34
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:35
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:36
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:36
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:37
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:39
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:39
▶ 155.33.133.46	/user/register/	mitigate-sqli	Block	00:42:39