Report: AWS WAF & Shield - Attack Vector Observations on Web Application

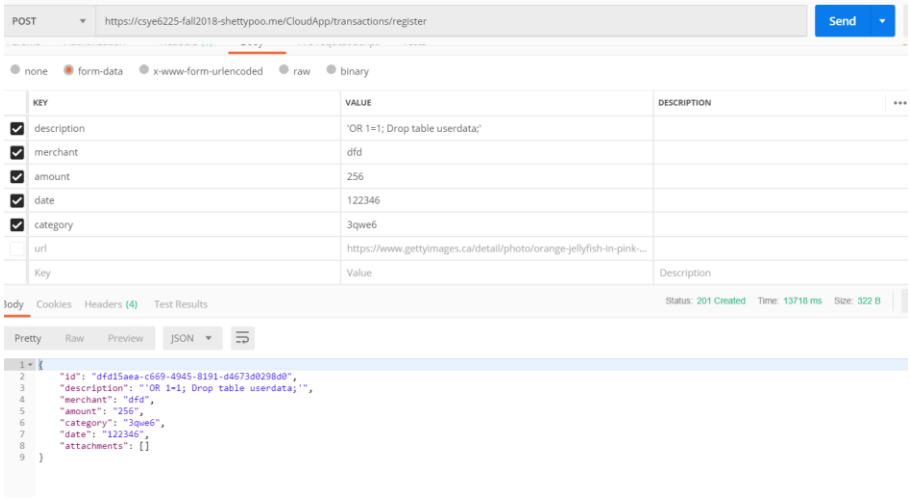
1. Attack Vector – SQL Injection:

a. *Why choose SQL injection?*

SQL injection attack gives insight for the attackers to manipulate and disrupt the existing data which will result in repudiation issues, including: modification to balances/transactions, releasing all data on system, and even destroying data. It can even result in attacker becoming the administrator of the database server.

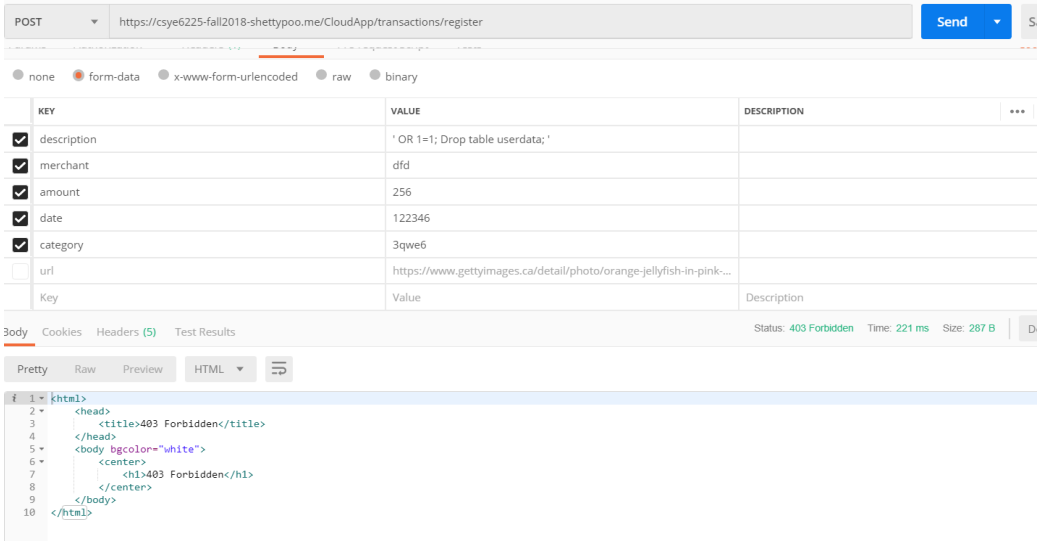
b. *Before implementing AWS WAF & Shield security*

In the screenshot below, it can be seen that the malicious data entered is stored in the application's database. The issue is that if this data is retrieved and utilized for inline SQL statement execution, there is a possibility that the table will be deleted. However, since the application we are dealing with is in Spring, it should not have any affect on the code.



c. *Testing/Results of AWS WAF and Shield*

The way the SQL injection scenario was tested can be seen in the screen shot below where the description is given value that is a command to drop the table. The result is that the Firewall identifies this and gives an output of 403 forbidden. Hence, there was no malicious data being stored in database.



Report: AWS WAF & Shield - Attack Vector Observations on Web Application

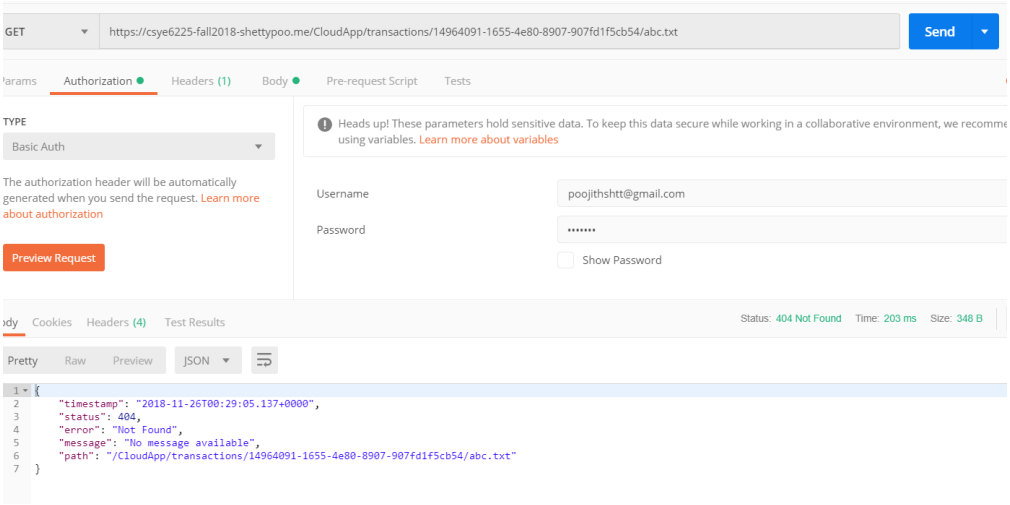
2. Attack Vector – Web Service File Access:

a. *Why choose web service file access?*

There can be attacks by having knowledge of the folder path of web service. This obviously is a violation since it allows the files to be vulnerable. The files may contain potential information, such as: username, password, etc. In order to prevent this, the proper steps should be taken to protect this file from the users in order to not leak the data.

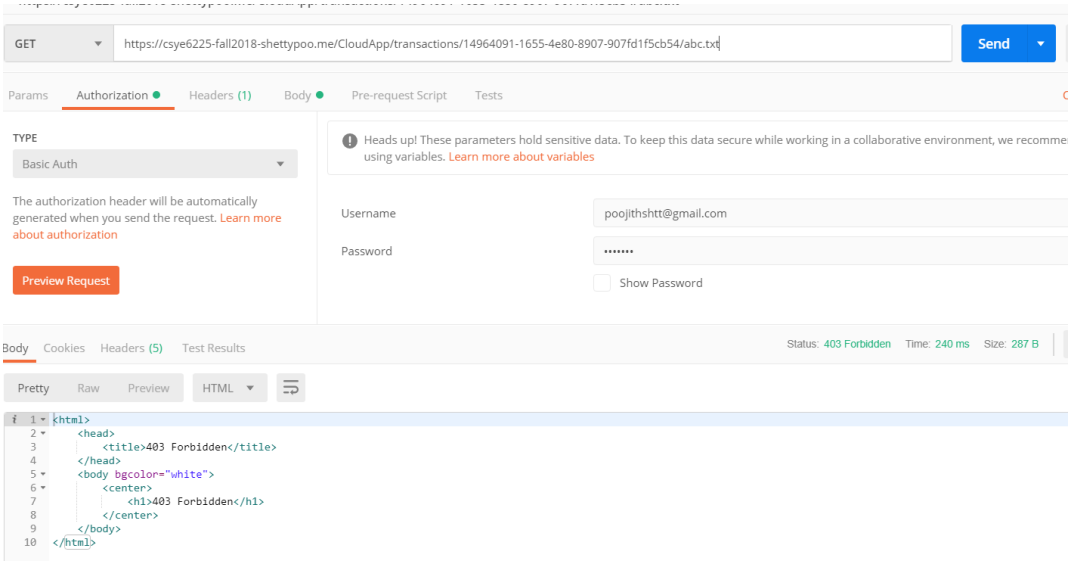
b. *Before implementing AWS WAF & Shield security*

In the below screenshot it can be clearly seen that during these kinds of requests, there is a hit to the application. For instance, when the web service contains any config file, the user will be able to access it by hitting the url path, and if it is a valid path, it will display the file.



c. *Testing/Results of AWS WAF and Shield*

After implementing the security for the load balancer, it can be seen that if the link hit has some sort of file, the AWS WAF and Shield will identify the files and restrict the user from accessing any sort of files from the client side.



Report: AWS WAF & Shield - Attack Vector Observations on Web Application

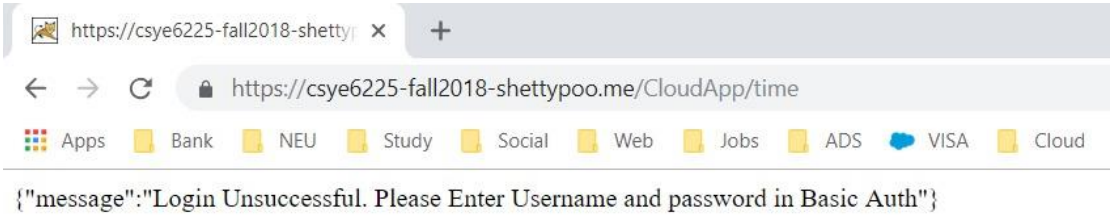
3. Attack Vector –IP Blacklist:

a. *Why choose IP Blacklist?*

Without restricting certain IP addresses, they have the ability to penetrate the resources and can manipulate it.

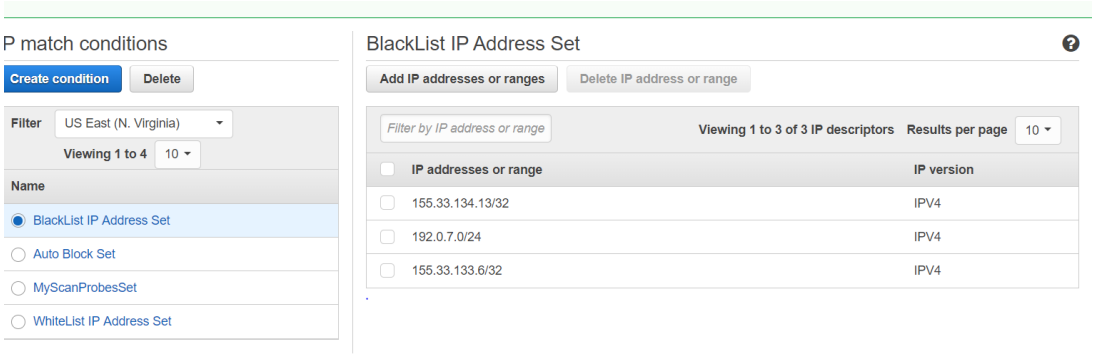
b. *Before implementing AWS WAF & Shield security*

In order to prevent certain IP addresses to access the application, it is possible to ad those IPs as part of the rules. Before these rules, all IP addresses were allowed to access the application.

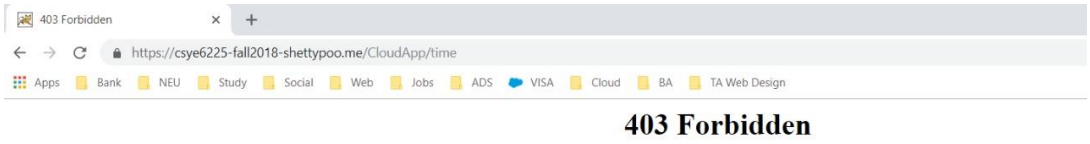


c. *Testing/Results of AWS WAF and Shield*

Using AWS WAF we can block some of the IP based on the requirement.
Currently we have added few IP addresses to the rule, which prevents the users with those particular IPs to not be able to access the web application.
Image below displays restricted IP address.



The image below is the result of a user accessing application with a blacklisted IP address.



Report: AWS WAF & Shield - Attack Vector Observations on Web Application

4. Attack Vector – URL Length Constraint:

a. *Why choose URL length constraint?*

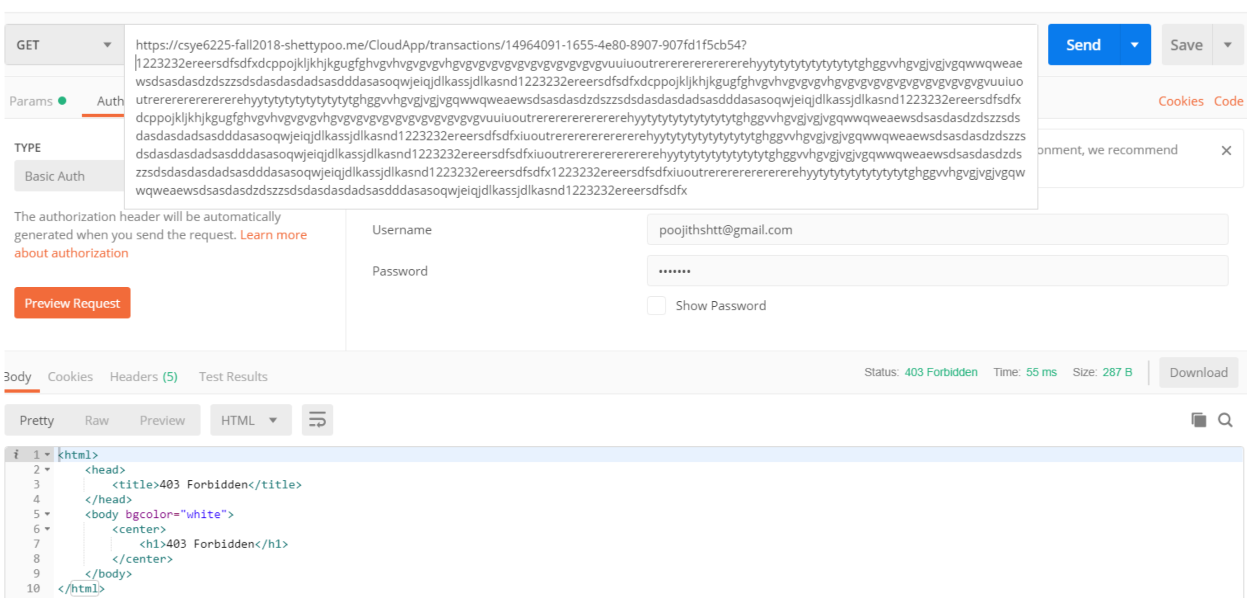
Under normal circumstances, the web application has a regular url which can be hit. Hence, the size of the url will be approximately known. However, the attackers will attempt to hit the address with a long url or invalid token in order to access the url.

b. *Before implementing AWS WAF & Shield security*

The url was hitting the web application which in turn responded by giving errors or results.

c. *Testing/Results of AWS WAF and Shield*

After implementation, the large URL is not permitted and is forbitten at the firewall level. This in turn stops all the bad requests at the beginning without the high load on the application.



Report: AWS WAF & Shield - Attack Vector Observations on Web Application

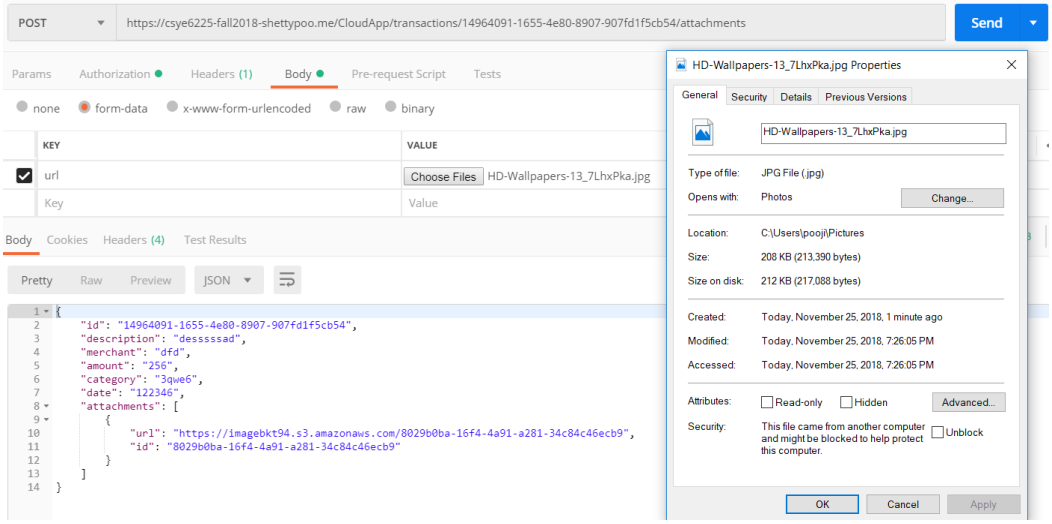
5. Attack Vector –Body Constraint:

a. *Why choose body constraint?*

When uploading files to the application, it has a possibility of risk. To start with, an attack will deploy some code to the system to be attacked. Then the only thing that is left is the code execution. This causes the system vulnerability through the ability for complete system takeover. Also, the large file upload can cause client-side attacks or even file system/database overload.

b. *Before implementing AWS WAF & Shield security*

Currently there is no restriction regarding the large image size which were added to the s3 bucket, therefore, if the user intends to fill the s3 bucket, it will crash the program.



c. *Testing/Results of AWS WAF and Shield*

Using AWS WAF, it is easy to add a constraint and keep the image size in check. The body is set to be less than 2MB, so the firewall will only allow the requests which has image size less than 2MB.