

CSYE 6225 : Network Structures and Cloud Computing

Description: This assignment aims at finding the security issues in the web application we built as part of previous assignments and try to come up with solutions for them.

Procedure: Try to come up with at least 3 attack vectors. Show the vulnerability in the application because of each attack vector. Install AWS Web Application Firewall (WAF) on the Load Balancer with correct AWS Rules to stop these type of attacks from happening.

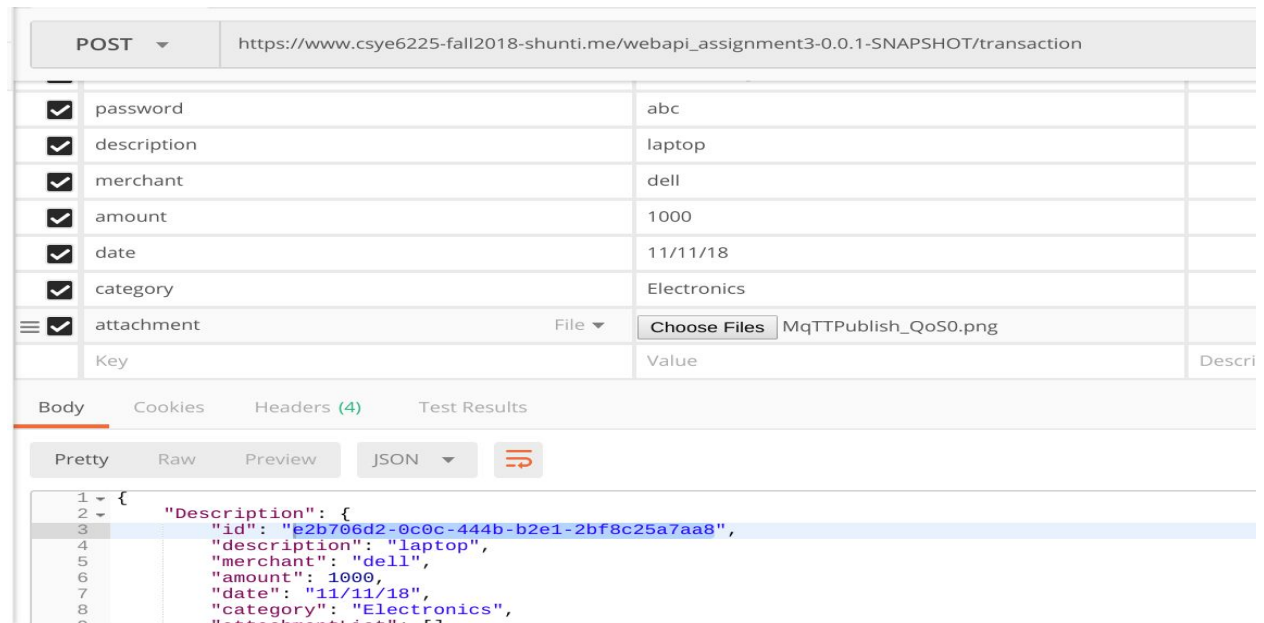
Test Cases:

CASE 1: Unlimited Size of Transaction Body

This is one of the most common vulnerability in a web application. Increase in the body size of a transaction needs exponential increase in processing speed of transactions and needs more storage space in the Data Base. Attackers can send huge number of false transaction requests in a small duration to cause the infrastructure to go out of processing capacity and storage capacity which may result in outage of service.

Our application was processing the transaction without considering the size of the transaction body (usually the attachment in the body can have huge size).

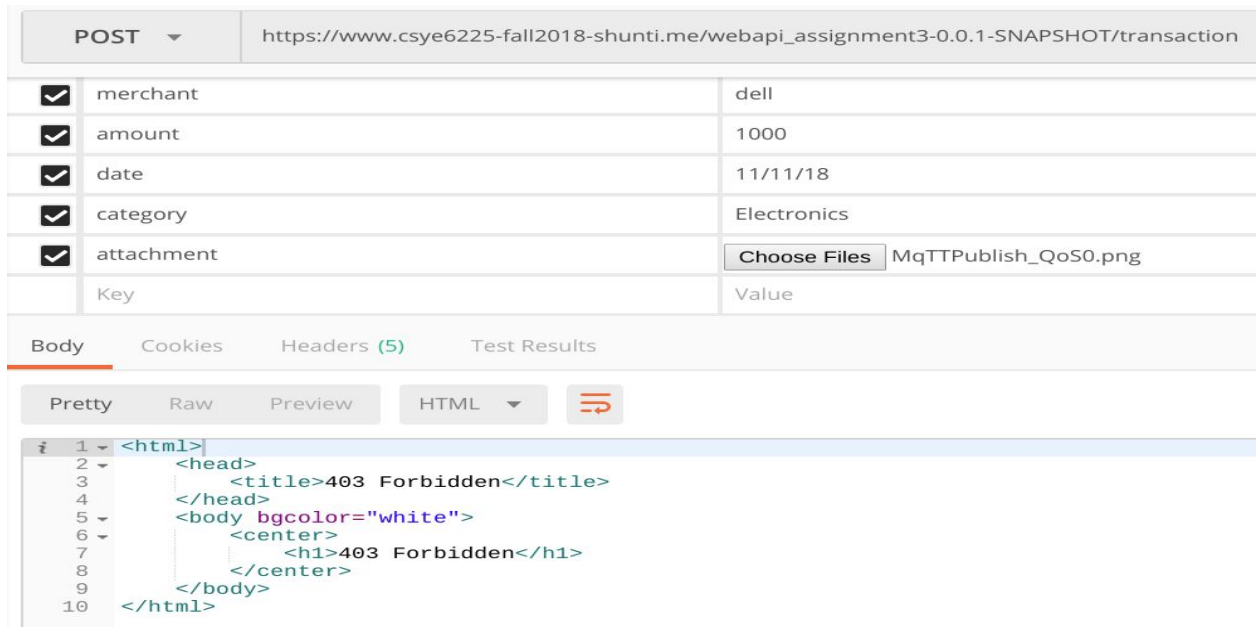
Tested the flaw in Postman:



Assignment 10

Mitigation: Installed WAF on the load balancer with a constraint which blocks the transaction if the body is more than 50kb.

When tested in Postman, the transaction did not go through:



CASE 2: Suspicious users

There are many attackers who are constantly looking to hack into applications and are constantly trying to send malicious data/requests to find out the vulnerability in the application. So these IP addresses should be monitored and blocked from doing any type of transaction with the application.

The malicious users can be blocked using AWS WAF, by setting IP addresses or ranges of IP addresses to the block list. Any request/transaction from these IPs are not processed.

Here, my IP address is 65.96.221.118/32 which is put under blocked IP list



Assignment 10

The transaction is clearly blocked from going through because of the firewall (New User registration is failed because the connection is forbidden).

POST https://www.csye6225-fall2018-shunti.me/webapi_assignment3-0.0.1-SNAPSHOT/user/save

form-data x-www-form-urlencoded raw binary

	KEY	VALUE	DE
<input checked="" type="checkbox"/>	username	newuser@gmail.com	
<input checked="" type="checkbox"/>	password	masdjn!	
	Key	Value	De

Body Cookies Headers (5) Test Results Statu

Pretty Raw Preview HTML

```
1 <html>
2   <head>
3     <title>504 Gateway Time-out</title>
4   </head>
5   <body bgcolor="white">
6     <center>
7       <h1>504 Gateway Time-out</h1>
8     </center>
9   </body>
10 </html>
```

CASE 3: SQL Injection:

A SQL Injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

We tried SQL injection technique using Automatic SQL injection and database takeover tool (as our application uses Hibernate I used SSH to connect to EC2 and then i tried to access the database).

Assignment 10

```
File Edit View Search Terminal Help
centos@ip-10-0-0-203:~/sqlmap-dev
[14:55:32] [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. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
[*] ending @ 14:55:32 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/transaction?id=b88bdd57-44bc-4102-9948-0cab96594038" --batch

(1.2.11.15sdev)
http://sqlmap.org

[!] 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 @ 15:01:09 /2018-11-25/

[15:01:09] [INFO] testing connection to the target URL
[15:01:09] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:01:10] [INFO] testing if the target URL content is stable
[15:01:11] [INFO] target URL content is stable
[15:01:12] [INFO] testing if GET parameter 'id' is dynamic
[15:01:12] [WARNING] GET parameter 'id' does not appear to be dynamic
[15:01:12] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[15:01:12] [INFO] testing for SQL injection on GET parameter 'id'
[15:01:12] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:01:12] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[15:01:16] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[15:01:17] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause'
[15:01:20] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[15:01:21] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[15:01:22] [INFO] testing 'MySQL >= 5.6 error-based - Parameter replace (FLOOR)'
[15:01:23] [INFO] testing 'MySQL inline queries'
[15:01:23] [INFO] testing 'PostgreSQL inline queries'
[15:01:23] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[15:01:23] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[15:01:23] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[15:01:23] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[15:02:30] [INFO] testing 'MySQL >= 5.6.12 AND time-based blind'
[15:02:31] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
there seems to be a continuous problem with connection to the target. Are you sure that you want to continue with further target testing? [y/N] N
it looks like the back-end runs in 'PostgreSQL'. 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 'PostgreSQL' extending provided level (1) and risk (1) values? [Y/n] Y
[15:11:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[15:11:00] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[15:11:20] [INFO] heuristics detected web page charset 'ascii'
[15:11:20] [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
[15:11:40] [INFO] target URL appears to have 39 column in query
[15:12:00] [WARNING] user aborted during detection phase
how do you want to proceed? [(S)kip current test/(e)xit detection phase/(n)ext parameter/(c)hange verbosity/(q)uit] q
[15:12:01] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 1 times, 502 (Bad Gateway) - 2 times
[15:12:07] [INFO] user quit
[*] ending @ 15:12:07 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/user/save?id=1" --batch

(1.2.11.15sdev)
http://sqlmap.org

[!] 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 @ 14:54:31 /2018-11-25/

[14:54:31] [INFO] testing connection to the target URL
[14:54:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[14:54:32] [INFO] testing if the target URL content is stable
[14:54:33] [INFO] target URL content is stable
[14:54:34] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1')
[*] ending @ 14:54:36 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/user/save?id=1" --batch

(1.2.11.15sdev)
http://sqlmap.org

[!] 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 @ 14:55:04 /2018-11-25/

[14:55:04] [INFO] testing connection to the target URL
[14:55:04] [INFO] testing if the target URL content is stable
[14:55:05] [INFO] target URL content is stable
[14:55:05] [INFO] testing if GET parameter 'id' is dynamic
[14:55:05] [WARNING] GET parameter 'id' does not appear to be dynamic
[14:55:05] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[14:55:05] [INFO] testing for SQL injection on GET parameter 'id'
[14:55:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:55:05] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:55:05] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:55:10] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:55:11] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:55:12] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:55:13] [INFO] testing 'MySQL >= 5.6 error-based - Parameter replace (FLOOR)'
[14:55:14] [INFO] testing 'MySQL inline queries'
[14:55:14] [INFO] testing 'PostgreSQL inline queries'
[14:55:14] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:55:14] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:55:15] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:55:15] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:55:15] [INFO] testing 'MySQL >= 5.6.12 AND time-based blind'
[14:55:17] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:55:17] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[14:55:20] [INFO] testing 'Oracle AND time-based blind'
[14:55:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:55:32] [WARNING] GET parameter 'id' does not seem to be injectable
[14:55:32] [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. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
[*] ending @ 14:55:32 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/transaction?id=b88bdd57-44bc-4102-9948-0cab96594038" --batch
```

After mitigating we got the Output cannot be injected:

```
File Edit View Search Terminal Help
centos@ip-10-0-0-203:~/sqlmap-dev
[!] 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 @ 14:54:31 /2018-11-25/

[14:54:31] [INFO] testing connection to the target URL
[14:54:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[14:54:32] [INFO] testing if the target URL content is stable
[14:54:33] [INFO] target URL content is stable
[14:54:34] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1')
[*] ending @ 14:54:36 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/user/save?id=1" --batch

(1.2.11.15sdev)
http://sqlmap.org

[!] 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 @ 14:55:04 /2018-11-25/

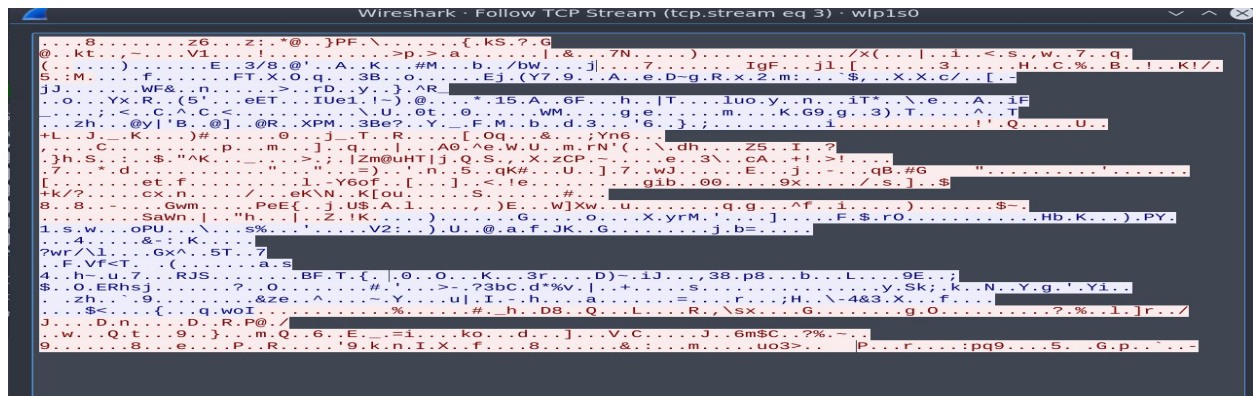
[14:55:04] [INFO] testing connection to the target URL
[14:55:04] [INFO] testing if the target URL content is stable
[14:55:05] [INFO] target URL content is stable
[14:55:05] [INFO] testing if GET parameter 'id' is dynamic
[14:55:05] [WARNING] GET parameter 'id' does not appear to be dynamic
[14:55:05] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[14:55:05] [INFO] testing for SQL injection on GET parameter 'id'
[14:55:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:55:05] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:55:05] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:55:10] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:55:11] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[14:55:12] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:55:13] [INFO] testing 'MySQL >= 5.6 error-based - Parameter replace (FLOOR)'
[14:55:14] [INFO] testing 'MySQL inline queries'
[14:55:14] [INFO] testing 'PostgreSQL inline queries'
[14:55:14] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:55:14] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:55:15] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:55:15] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:55:15] [INFO] testing 'MySQL >= 5.6.12 AND time-based blind'
[14:55:17] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:55:17] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[14:55:20] [INFO] testing 'Oracle AND time-based blind'
[14:55:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:55:32] [WARNING] GET parameter 'id' does not seem to be injectable
[14:55:32] [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. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
[*] ending @ 14:55:32 /2018-11-25/

[centos@ip-10-0-0-203 sqlmap-dev]$ python sqlmap.py -u "https://csye6225-fall2018-joshisuj.me:80/webapi_assignment3-0.0.1-SNAPSHOT/transaction?id=b88bdd57-44bc-4102-9948-0cab96594038" --batch
```

Additional Security Checks:

We have also implemented the basic security constraints in the web application like SSL over HTTP, which will make the packets not decryptable. And the intercepted packets cannot be interpreted.

TCP packet data (user/save API) captured in WireShark:



Also we have only 1 secure port open:

NMap screenshot:

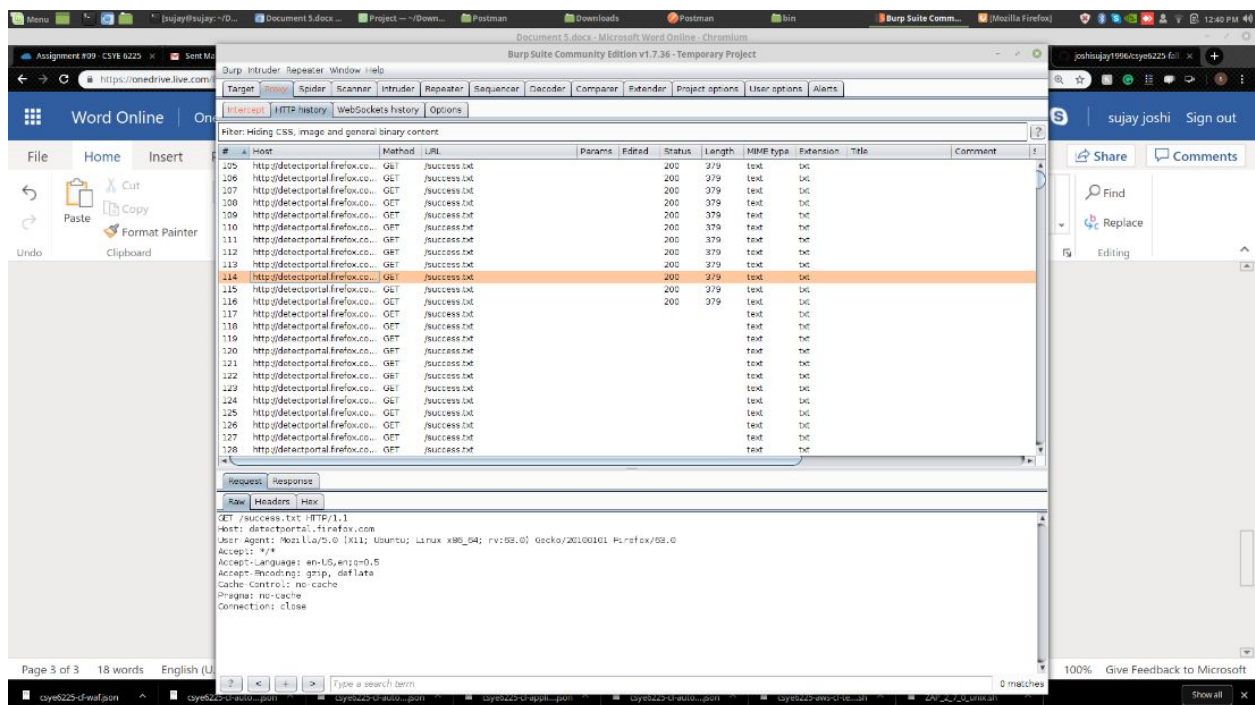
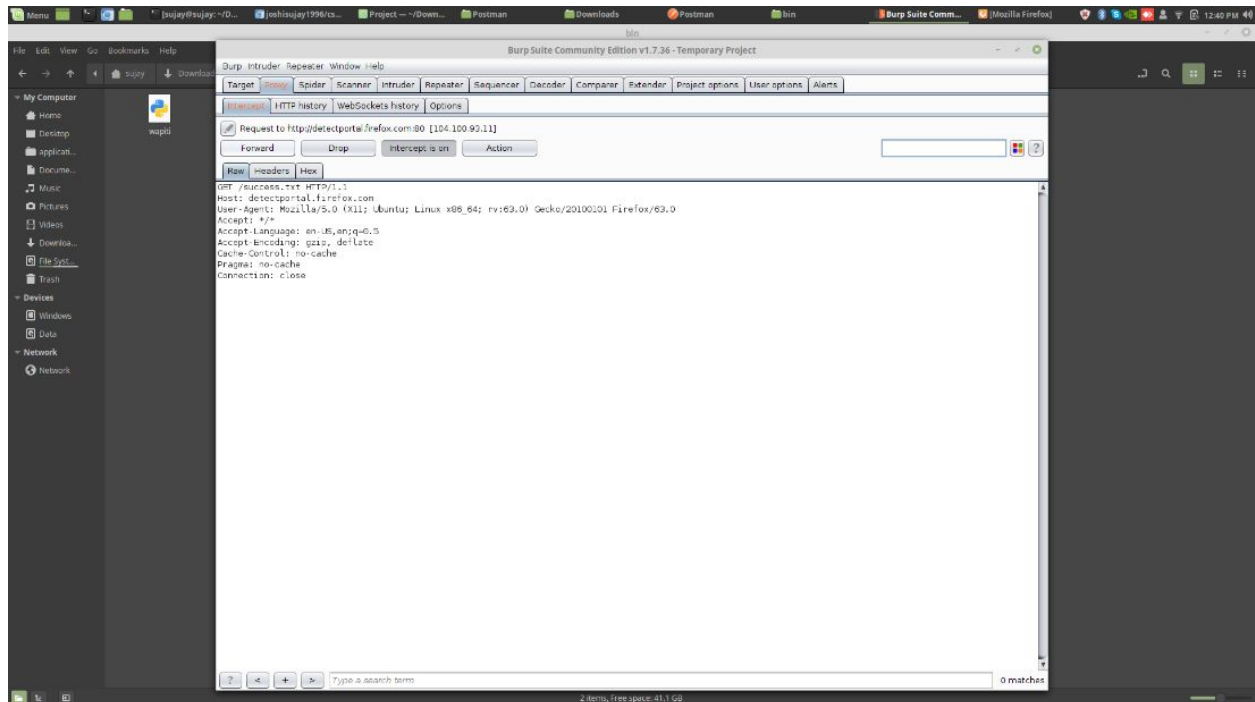
```
[ashwath@ashwathShunti ~]$ nmap www.csye6225-fall2018-shahdhw.me

Starting Nmap 7.60 ( https://nmap.org ) at 2018-11-26 17:39 EST
Nmap scan report for www.csye6225-fall2018-shahdhw.me (18.207.16.220)
Host is up (0.034s latency).
Other addresses for www.csye6225-fall2018-shahdhw.me (not scanned): 34.195.225.88
rDNS record for 18.207.16.220: ec2-18-207-16-220.compute-1.amazonaws.com
Not shown: 995 filtered ports
PORT      STATE SERVICE
22/tcp    closed ssh
80/tcp    closed http
443/tcp   open  https
3306/tcp  closed mysql
8080/tcp  closed http-proxy
```

We have also tested Cross Site forgery request test using Brup Suite.

The application is secure because of the https connection:

Assignment 10



Conclusion: With the installation of AWS Web Application Firewall and SSL protocol implementation, we can conclude that the application is reasonably safe from attacks.