NETWORK STRUCTURE AND CLOUD COMPUTING (CSYE-6225)

PENETRATION TESTING

ATTACK VECTORS

Introduction

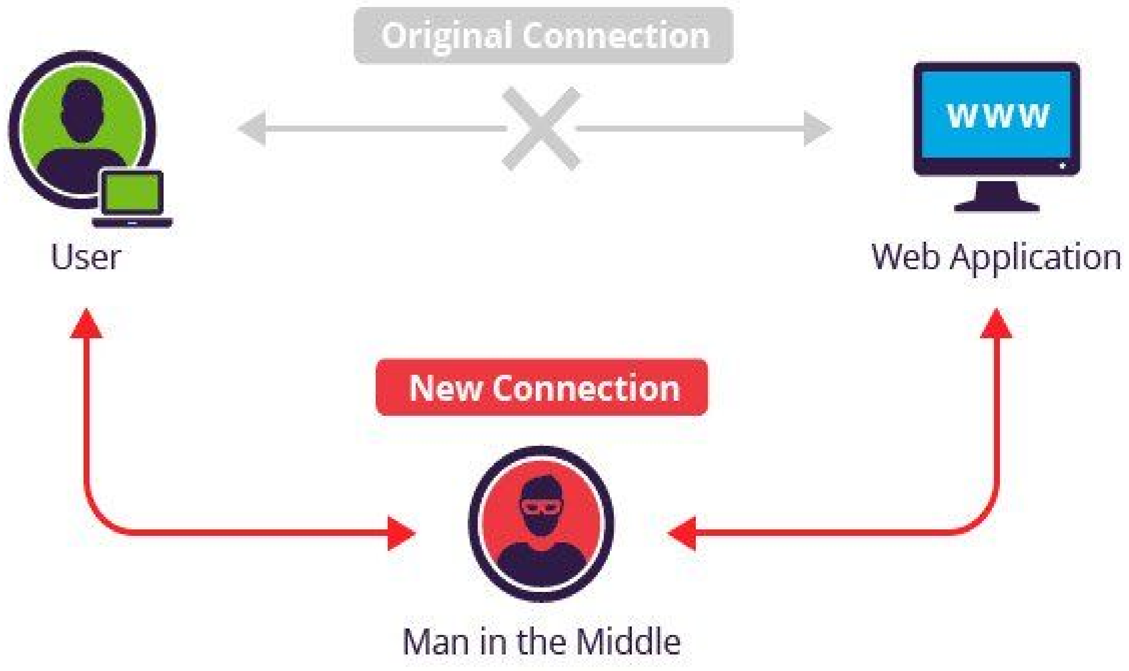
This document details the attack vectors used to perform penetration testing on an application that can be accessed via Restful APIs hosted on an EC2 instance. The Application is can register a user. The primary purpose of the application is for users to keep track of their expenditure by also giving an option to upload receipts to a expense tracked. A registered user can perform CRUD operations on a transaction that he created as well as attach receipts to that transaction. There is also an option to reset the user password and for a registered user to fetch the current time by providing authentication in the request header.

Below are Attack Vectors used to perform penetration testing on the application.

Attack Vectors

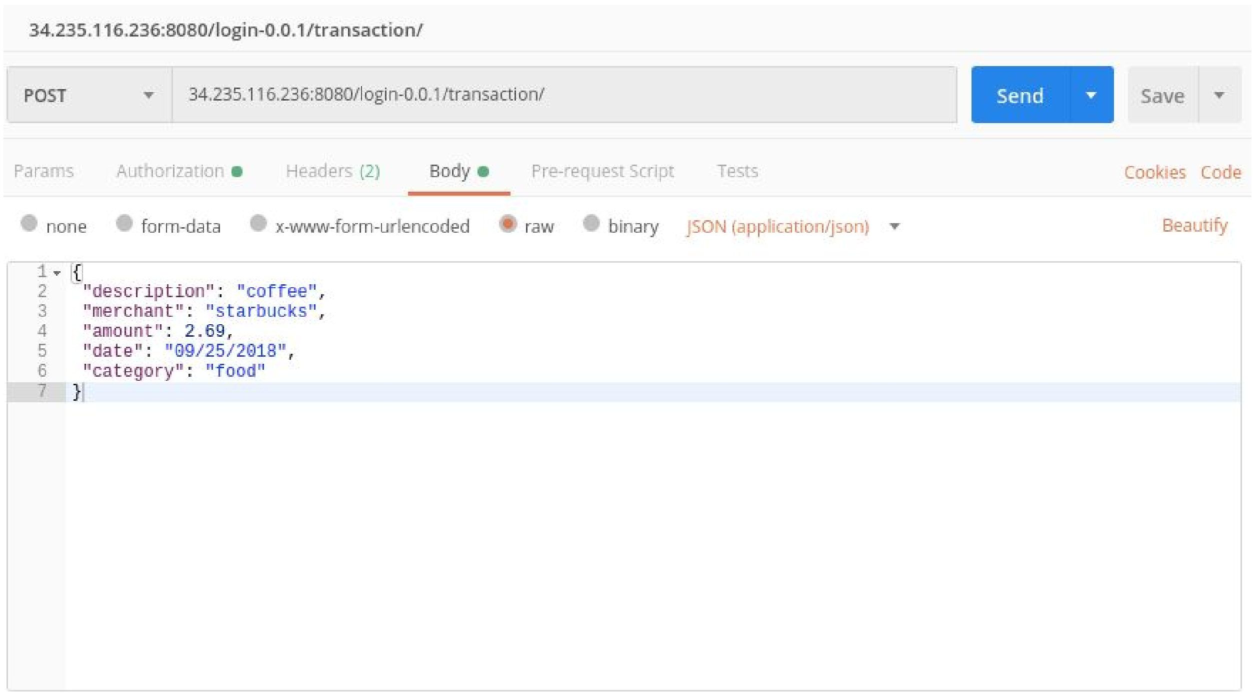
1. Man in the middle attack (MITM):

The Man in the middle attack is a term used when a hacker intercepts request from the user or the client which is the browser in most cases and the server where the application that servers the request resides.



In our scenario, this attack was used while creating a transaction using a POST request. The registered user attempts to create a transaction with a specific amount of $2.69. This request is intercepted by the hacker and the amount is changed from $2.69 to $3000 and sent ahead to the server. The server has no idea of the modified request and treats it as the original request and thus saves the transaction data for that particular user. The pictorial description of this attacks is as shown below.

The Original Request :

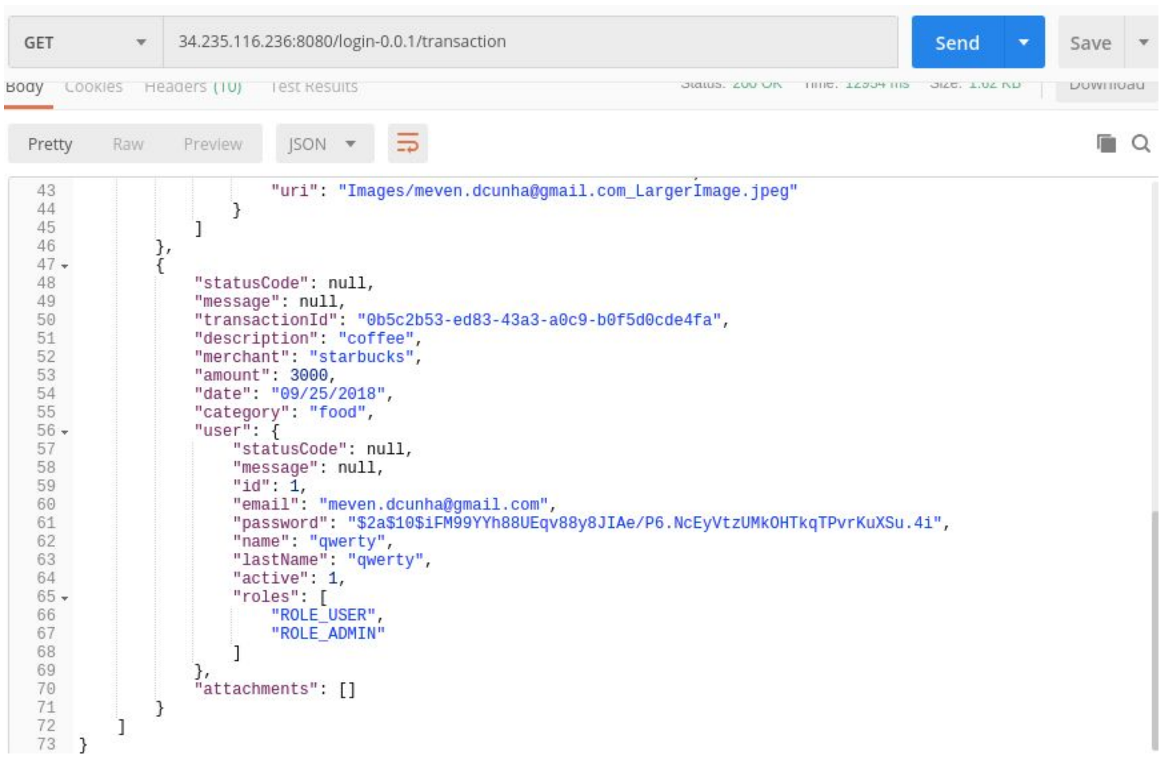

34.235.116.236:8080/login-0.0.1/transaction/

```
{
  "description": "coffee",
  "merchant": "starbucks",
  "amount": 2.69,
  "date": "09/25/2018",
  "category": "food"
}
```

The Modified Request :

The user on fetching his transactions would now see the modified transaction in the response

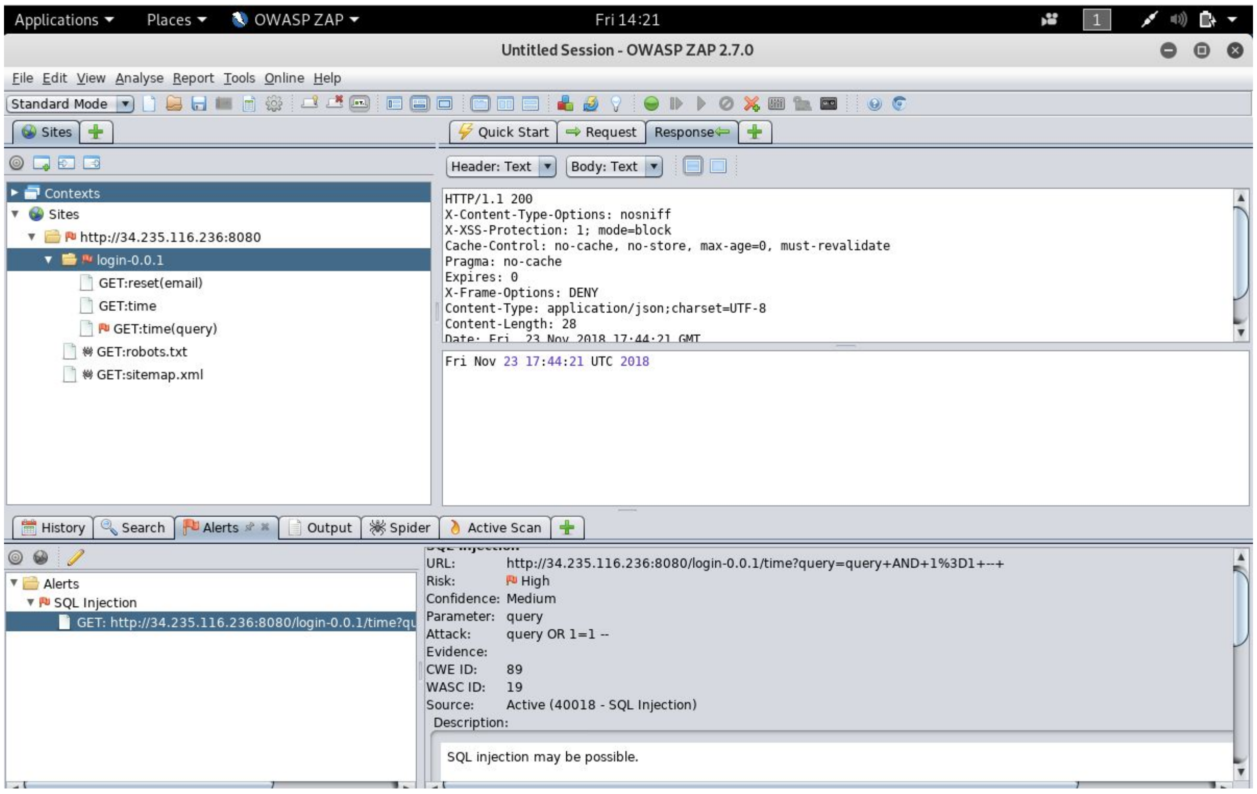
The Edited Response :



```
                    "uri": "Images/meven.dcunha@gmail.com_LargerImage.jpeg"
                }
            ]
        },
        {
            "statusCode": null,
            "message": null,
            "transactionId": "0b5c2b53-ed83-43a3-a0c9-b0f5d0cde4fa",
            "description": "coffee",
            "merchant": "starbucks",
            "amount": 3000,
            "date": "09/25/2018",
            "category": "food",
            "user": {
                "statusCode": null,
                "message": null,
                "id": 1,
                "email": "meven.dcunha@gmail.com",
                "password": "$2a$10$iFM99YYh88UEqv88y8JIAe/P6.NcEyVtzUMkOHTkqTPvrKuXSu.4i",
                "name": "qwerty",
                "lastName": "qwerty",
                "active": 1,
                "roles": [
                    "ROLE_USER",
                    "ROLE_ADMIN"
                ]
            },
            "attachments": []
        }
    ]
}
```

2 . SQL Injection Attack

This attack uses malicious SQL code to manipulate the backend database information and could also lead to loss of data altogether. In our specific scenario we have used the tool OWASP ZAP to check for SQL injection vulnerabilities in the application. The report generated is as shown below



```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Content-Length: 28
Date: Fri, 23 Nov 2018 17:44:21 GMT
```

```
Fri Nov 23 17:44:21 UTC 2018
```

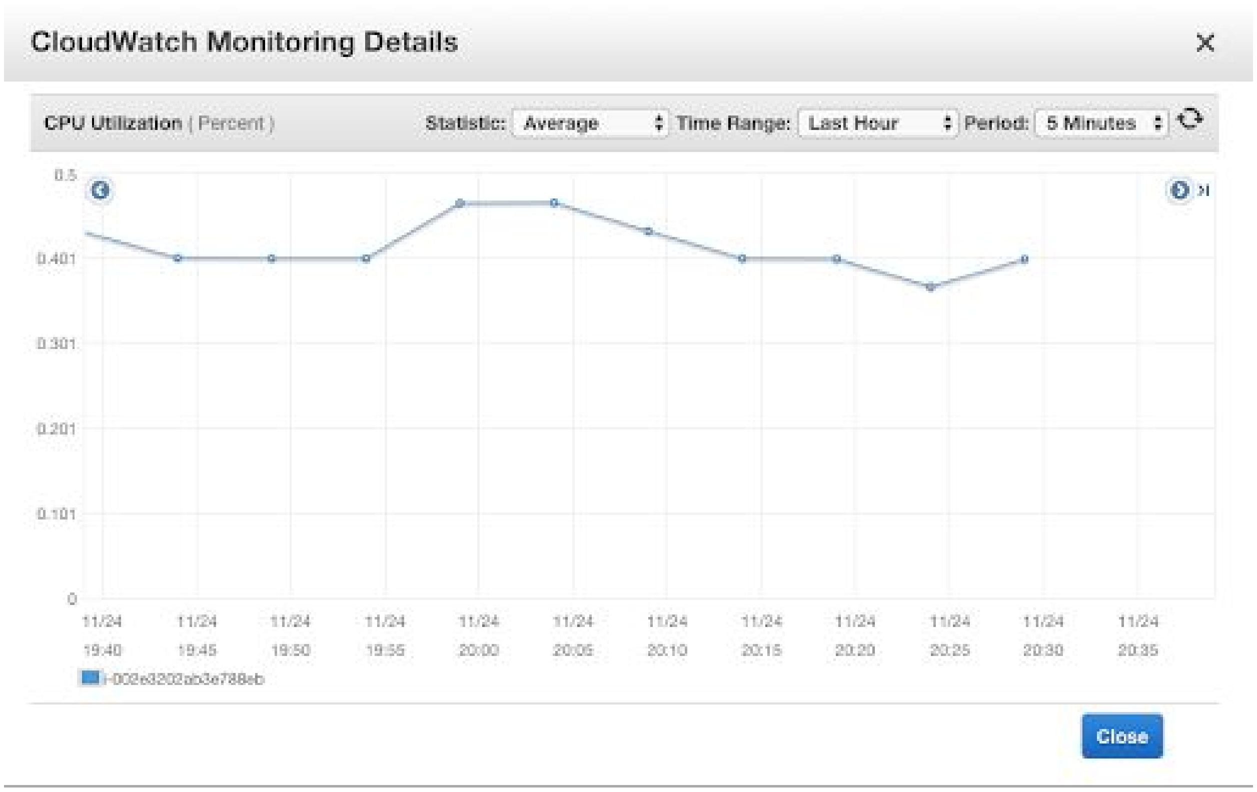
```
URL:         http://34.235.116.236:8080/login-0.0.1/time?query=query+AND+1%3D1+--+
Risk:        High
Confidence:  Medium
Parameter:   query
Attack:      query OR 1=1 --
Evidence:
CWE ID:      89
WASC ID:     19
Source:      Active (40018 - SQL Injection)
 Description:

 SQL injection may be possible.
```

From the above result it can be seen that an SQL injection can be possible on the get time API for a registered user. Although the chances of attempting a SQL injection in a GET request seems minimal, the report suggest that with the api call modification as shown in the image, there is a possibility of an SQL injection and has termed it as a High risk alert.

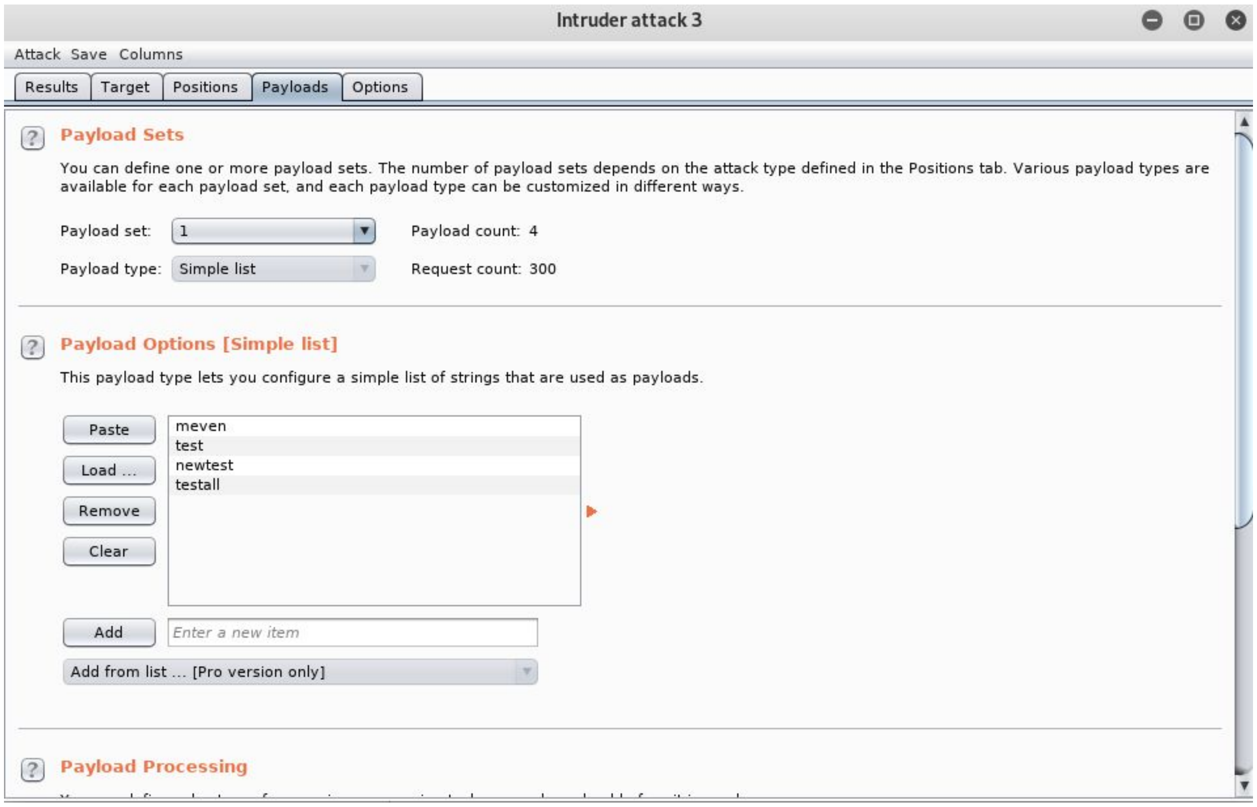
3. Brute Force Attack to degrade performance

Brute Force attack is typically used to gain access to username and password by repeatedly attempting various combinations. The way Brute Force has been used in our scenario is a bit deviant in our scenario. The reset password API is repeatedly hit (3000 request) thus impacting the performance of the instance. The graph of the performance when the attack was attempted is as shown below :



From the above image it can be noted that the CPU Utilization of the instance increased by approximately 8% when the Brute force attack was attempted. The main goal of the hack here is to degrade the server utilization eventually.

For the above reading the Brute force attack was made for 300 request using Burp Suite which is depicted in the below images



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 4

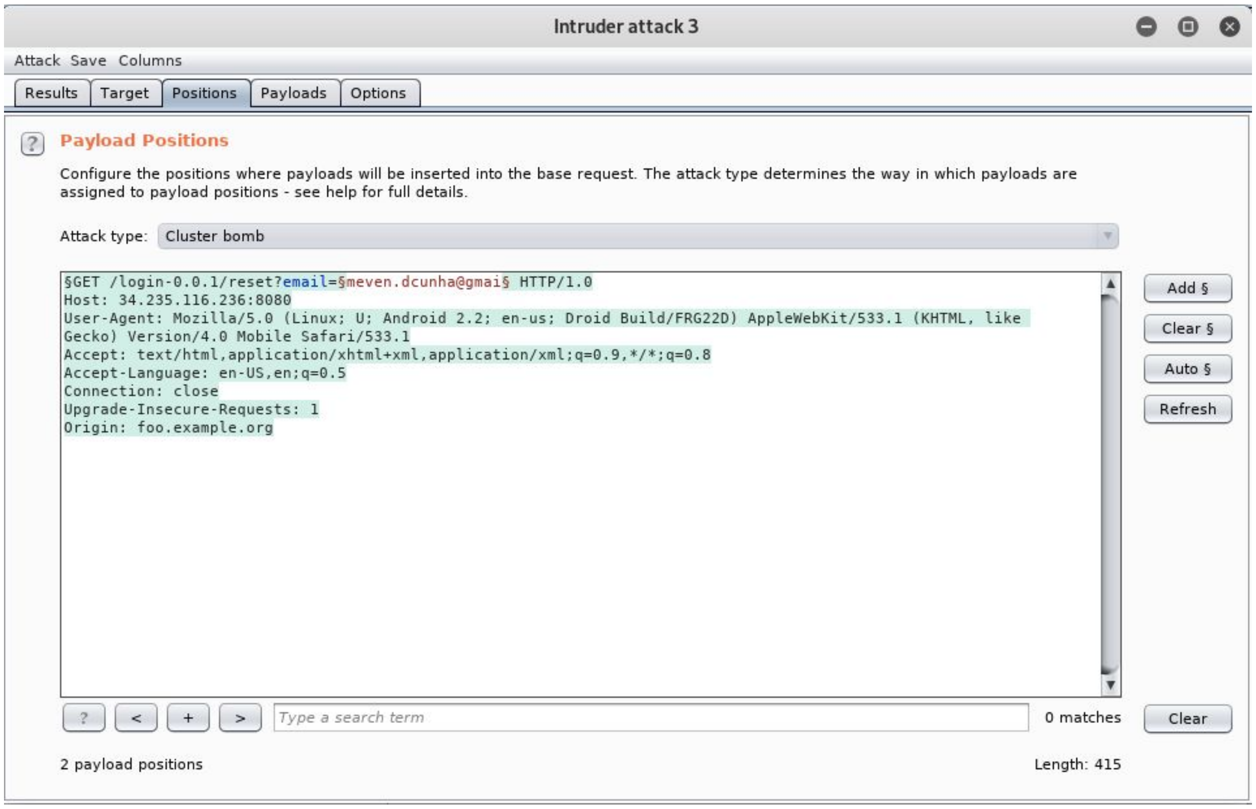
Payload type: Simple list Request count: 300

Payload Options [Simple list]

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

```
meven
test
newtest
testall
```

Payload Processing



Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

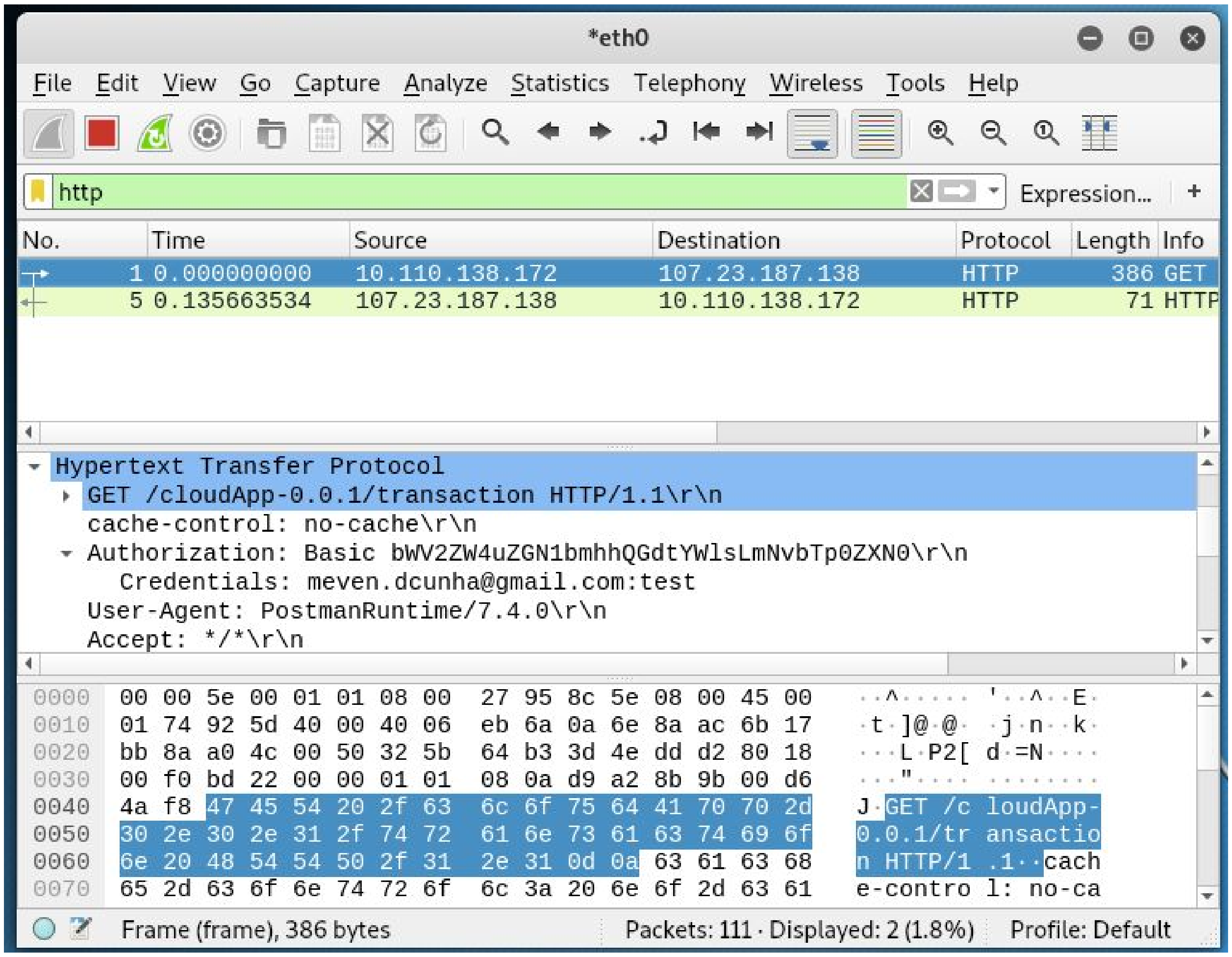
Attack type: Cluster bomb

```
§GET /login-0.0.1/reset?email=§meven.dcunha@gmai§ HTTP/1.0
Host: 34.235.116.236:8080
User-Agent: Mozilla/5.0 (Linux; U; Android 2.2; en-us; Droid Build/FRG22D) AppleWebKit/533.1 (KHTML, like
Gecko) Version/4.0 Mobile Safari/533.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Upgrade-Insecure-Requests: 1
Origin: foo.example.org
```

2 payload positions Length: 415

4. Sensitive Data Exposure

The Wireshark application is used to intercept the Basic Authentication in the request and fetch the username and password of the user. The image below describes this situation.


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...

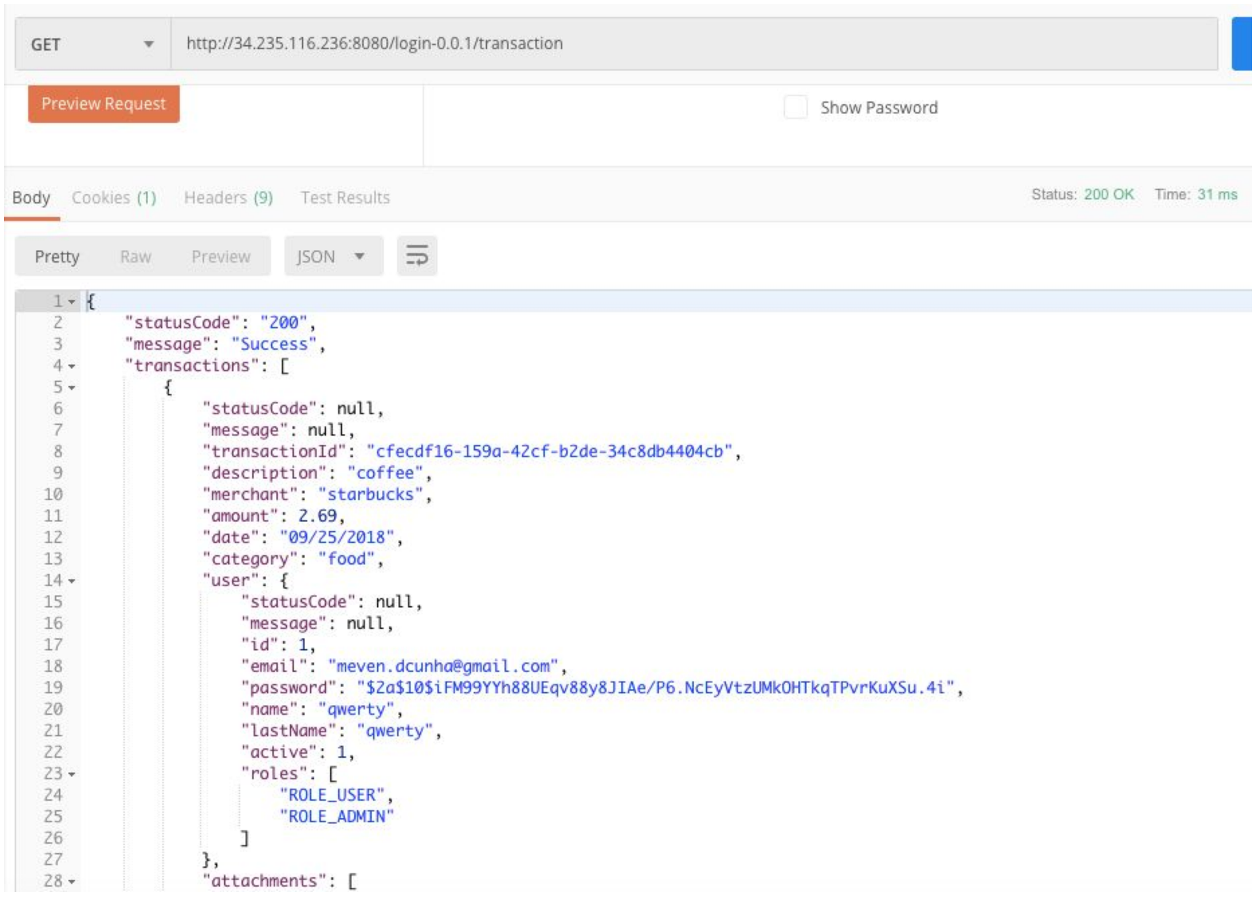
```
No.  Time         Source          Destination     Protocol  Length Info
   1 0.000000000  10.110.138.172  107.23.187.138  HTTP         386 GET
   5 0.135663534  107.23.187.138  10.110.138.172  HTTP          71 HTTP
```

```
Hypertext Transfer Protocol
  GET /cloudApp-0.0.1/transaction HTTP/1.1\r\n
  cache-control: no-cache\r\n
  Authorization: Basic bWV2ZW4uZGN1bmhhQGdtYWlsLmNvbTp0ZXN0\r\n
    Credentials: meven.dcunha@gmail.com:test
  User-Agent: PostmanRuntime/7.4.0\r\n
  Accept: */*\r\n
```

```
0000  00 00 5e 00 01 01 08 00  27 95 8c 5e 08 00 45 00   ··^·····  '··^··E·
0010  01 74 92 5d 40 00 40 06  eb 6a 0a 6e 8a ac 6b 17   ·t·]@·@·  ·j·n··k·
0020  bb 8a a0 4c 00 50 32 5b  64 b3 3d 4e dd d2 80 18   ···L·P2[  d·=N····
0030  00 f0 bd 22 00 00 01 01  08 0a d9 a2 8b 9b 00 d6   ···"····  ········
0040  4a f8 47 45 54 20 2f 63  6c 6f 75 64 41 70 70 2d   J·GET /c  loudApp-
0050  30 2e 30 2e 31 2f 74 72  61 6e 73 61 63 74 69 6f   0.0.1/tr  ansactio
0060  6e 20 48 54 54 50 2f 31  2e 31 0d 0a 63 61 63 68   n HTTP/1  .1··cach
0070  65 2d 63 6f 6e 74 72 6f  6c 3a 20 6e 6f 2d 63 61   e-contro  l: no-ca
```

Frame (frame), 386 bytes · Packets: 111 · Displayed: 2 (1.8%) · Profile: Default

This attack can be triggered when some sensitive data is revealed in the application. In the case of our Rest application, on fetching the transaction the user data is revealed for who that transaction belongs to. Below is the image that demonstrates that.



http://34.235.116.236:8080/login-0.0.1/transaction

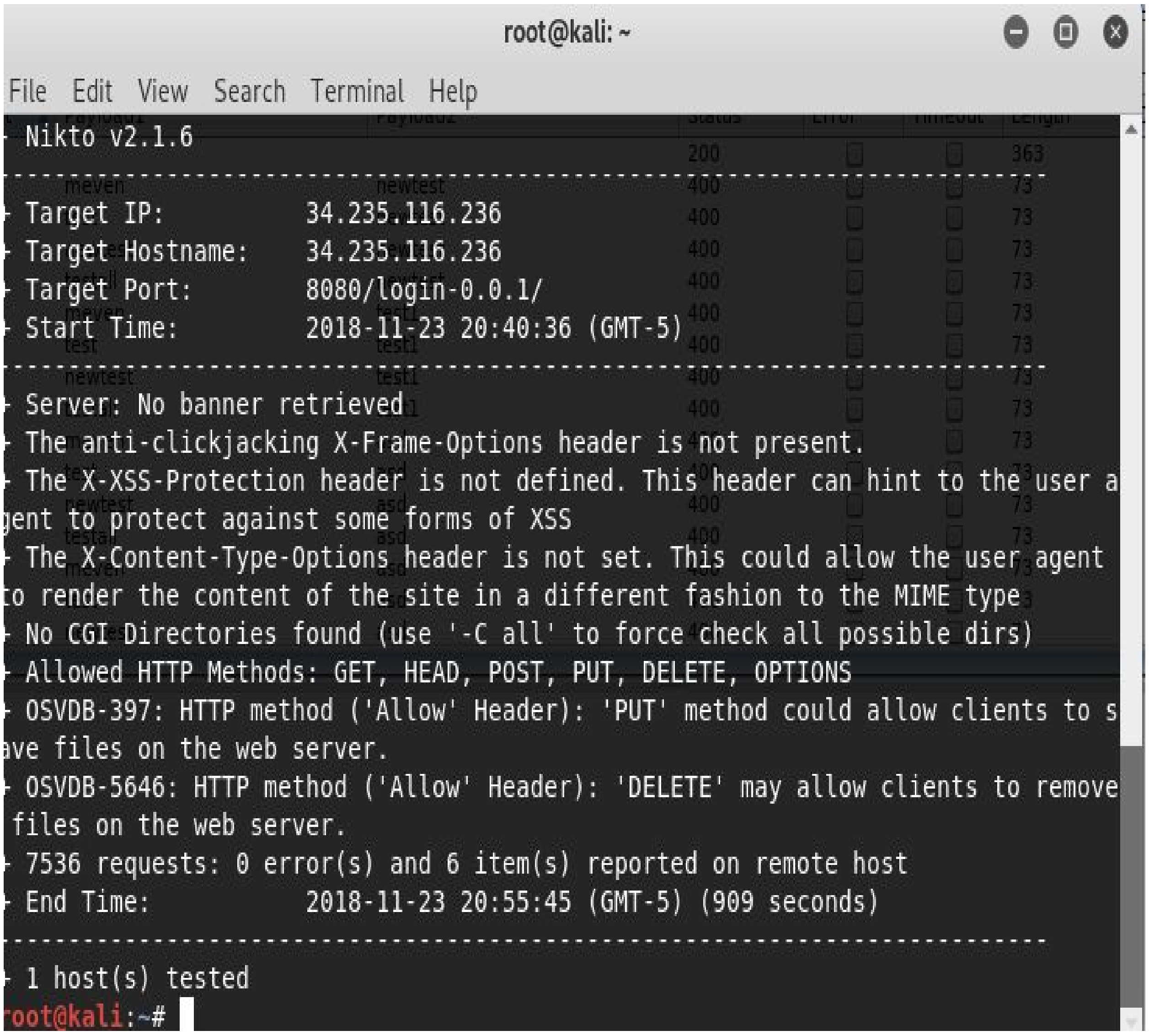
Status: 200 OK Time: 31 ms

```
{
    "statusCode": "200",
    "message": "Success",
    "transactions": [
        {
            "statusCode": null,
            "message": null,
            "transactionId": "cfecdf16-159a-42cf-b2de-34c8db4404cb",
            "description": "coffee",
            "merchant": "starbucks",
            "amount": 2.69,
            "date": "09/25/2018",
            "category": "food",
            "user": {
                "statusCode": null,
                "message": null,
                "id": 1,
                "email": "meven.dcunha@gmail.com",
                "password": "$2a$10$iFM99YYh88UEqv88y8JIAe/P6.NcEyVtzUMkOHTkqTPvrKuXSu.4i",
                "name": "qwerty",
                "lastName": "qwerty",
                "active": 1,
                "roles": [
                    "ROLE_USER",
                    "ROLE_ADMIN"
                ]
            },
            "attachments": [
```

This attack when combined with the Man in the middle attack and provide the hacker with user data. Although the password stored in this data is encrypted by a unique SALT using Bcrypt, by using intensive resources to figure out the SALT used for a particular user, this could be decrypted. Also, the email Id intercepted would give the hacker information of the registered users.

5. Unprotected APIs

By using Nikto scanner, the potential vulnerabilities in the APIs were detected. The image of the scanned result is as shown below



```
 Nikto v2.1.6
---------------------------------------------------------------------------
 Target IP:          34.235.116.236
 Target Hostname:    34.235.116.236
 Target Port:        8080/login-0.0.1/
 Start Time:         2018-11-23 20:40:36 (GMT-5)
---------------------------------------------------------------------------
 Server: No banner retrieved
 The anti-clickjacking X-Frame-Options header is not present.
 The X-XSS-Protection header is not defined. This header can hint to the user a
gent to protect against some forms of XSS
 The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
 No CGI Directories found (use '-C all' to force check all possible dirs)
 Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
 OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to s
ave files on the web server.
 OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove
 files on the web server.
 7536 requests: 0 error(s) and 6 item(s) reported on remote host
 End Time:           2018-11-23 20:55:45 (GMT-5) (909 seconds)
---------------------------------------------------------------------------
 1 host(s) tested
oot@kali:~#
```

All of the attacks apart for the 4th one were prevented using the Web Application Firewall (WAF) with OWASP’s Top 10 Web Application Vulnerabilities configured.