

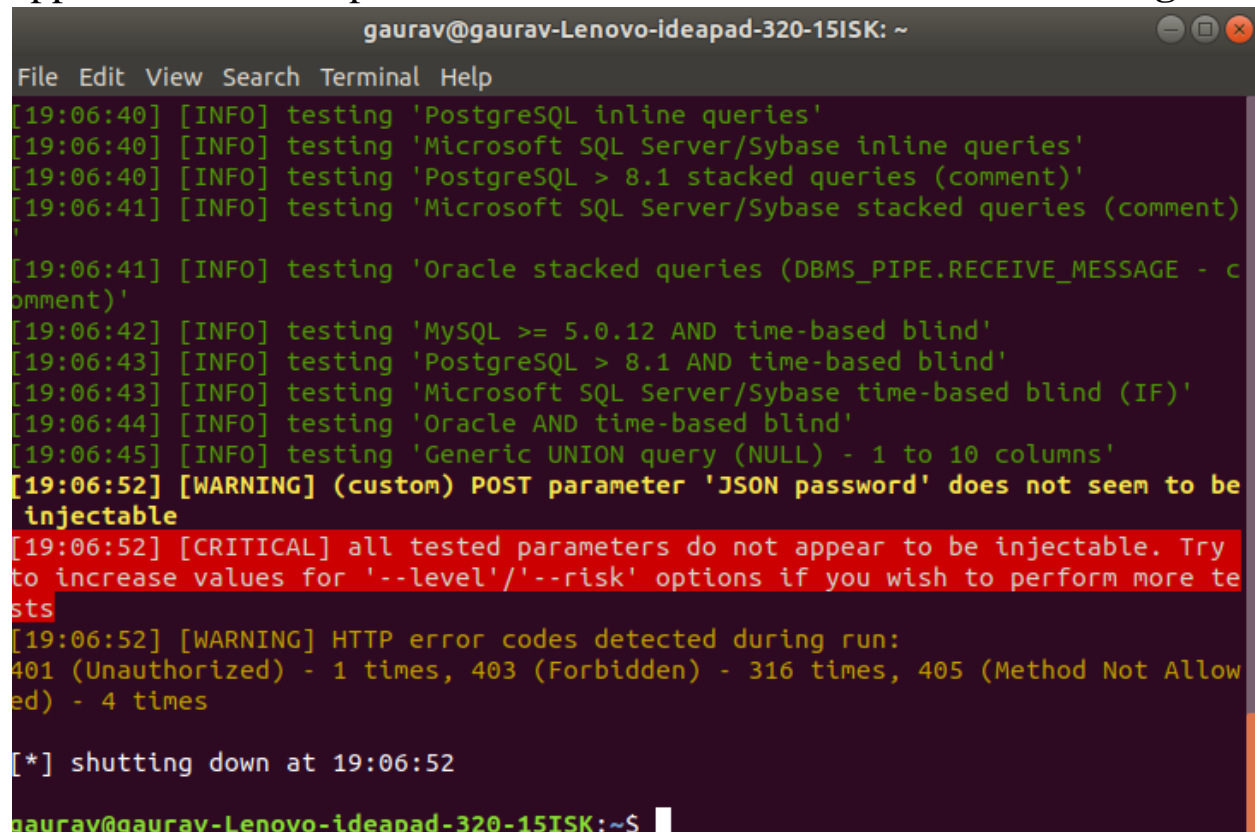
## Introduction

This document focuses on the attack vectors that are used to penetrate through the RESTFUL API's on a cloud EC2 instance. We are trying to register a user and authenticate the login process. The vulnerabilities of the web application can be protected using a firewall. These attacks mentioned below occur when the application is not protected from by the firewall.

## Attack Vectors:

### SQL Injection:

SQL injection happens when the attacker inputs malicious code in the backend database accordingly for personal interest. We have used SQL Map in the AWS firewall manager to check SQL vulnerabilities in the applications. The report is as follows: **After the firewall is running:**

A screenshot of a terminal window titled 'gaurav@gaurav-Lenovo-ideapad-320-15ISK: ~'. The terminal shows the output of a SQLMap test. It lists several tests for different database systems and query types, all marked as '[INFO]'. A warning message at 19:06:52 states that the 'JSON password' parameter is not injectable. A critical message at 19:06:52 states that all tested parameters do not appear to be injectable. A warning message at 19:06:52 lists HTTP error codes detected: 401 (Unauthorized) - 1 times, 403 (Forbidden) - 316 times, 405 (Method Not Allowed) - 4 times. The terminal ends with a message '[\*] shutting down at 19:06:52' and the prompt 'gaurav@gaurav-Lenovo-ideapad-320-15ISK:~\$'.

The SQL injection In a Post request shows that the values are injectable.

## Size Constraints:

The size constraints in the amazon web services firewall manager console gives us an option to make a constraint on the size of the attachment a user can send via the web into the s3 bucket.

### When sending the attachment > 1048576 bytes that is almost 1MB

The screenshot shows a REST client interface with the following details:

- URL:** `https://csye6225-su19-thakurga.me/book/e9caf1fa-b34b-4bcd-81b5-b8649a90a052/image`
- Method:** POST
- Body Type:** form-data (selected)
- Body Content:** A table with one row: 

KEY	VALUE	DESCRIPTION
file	<input type="button" value="Choose Files"/> test.jpg	
- Status:** 200 OK
- Time:** 786 ms
- Size:** 510 B
- Response Body (JSON):**

```
{  "id": "e9caf1fa-b34b-4bcd-81b5-b8649a90a052",  "uri": "https://csye6225-su19-thakurga.me.csye6225.com/s3.amazonaws.com/CoverImage/e9caf1fa-b34b-4bcd-81b5-b8649a90a052_test.jpg"}
```

### When sending the attachment < 1048576 bytes

The screenshot shows a REST client interface with the following details:

- URL:** `https://csye6225-su19-thakurga.me/book/62b63487-5949-4b11-8357-ea2718fb5371/image`
- Method:** POST
- Body Type:** form-data (selected)
- Body Content:** A table with one row: 

KEY	VALUE	DESCRIPTION
test	<input type="button" value="Choose Files"/> test.jpg	
- Status:** 500 Internal Server Error
- Time:** 718 ms
- Size:** 569 B
- Response Body (JSON):**

```
{  "timestamp": "2019-08-11T00:01:28.493+0000",  "status": 500,  "error": "Internal Server Error",  "message": "Maximum upload size exceeded; nested exception is java.lang.IllegalStateException: org.apache.tomcat.util.http.fileupload.UploadBase$FileSizeLimitExceededException: The field test exceeds its maximum permitted size of 1048576 bytes.",  "path": "/book/62b63487-5949-4b11-8357-ea2718fb5371/image"}
```