Assignment #08

CSYE 6225 : Network Structures and Cloud Computing

Aim : To find the security issues in web application developed
as a part of previous assignments and finding appropriate solutions for the same.

Procedure : Working around with minimum 3 attack vectors. Figure out and solve vulnerability in the application because of each attack vector. Installing AWS Web Application Firewall (WAF)on the Load Balancer applying required AWS Rules to prevent these type of attacks from happening.

Test Cases :

Case 1: Large Size of Attachment Body

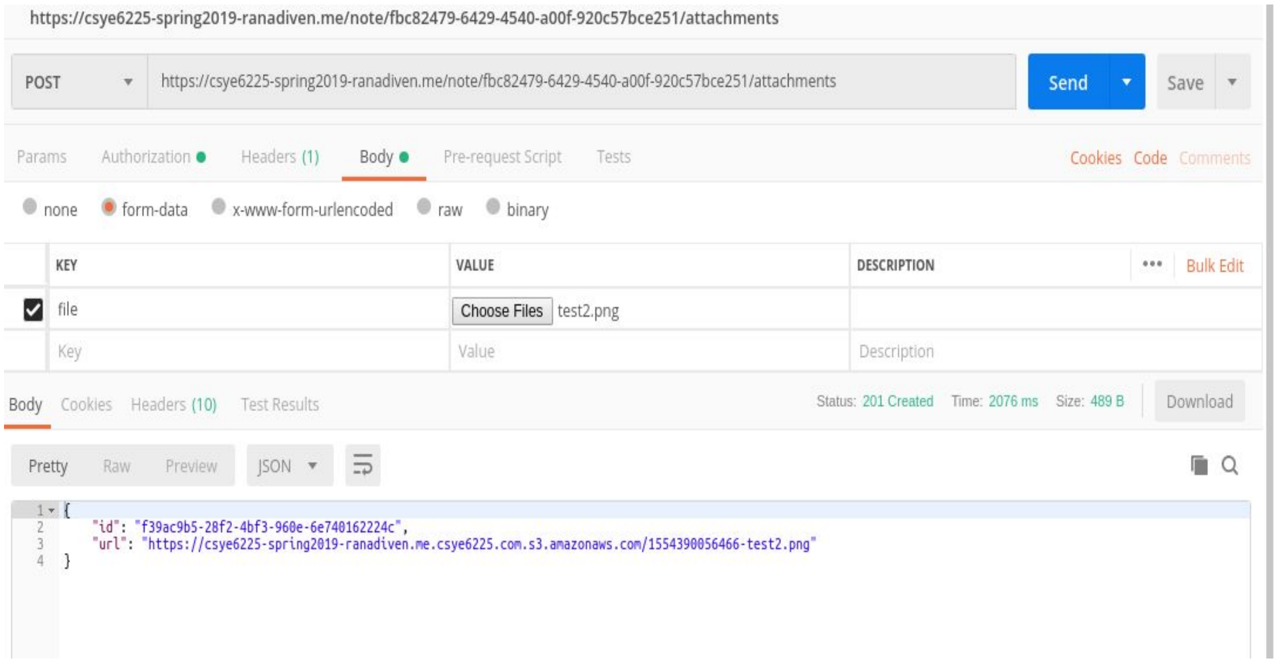
The body size of an attachment can be a vulnerability in any web application as when the body size increases it results in higher requirement of storage space and processing speed. If high number of requests are sent by attackers in a short amount of time it could cause service outage if our processing and storage capacity goes beyond what is available in our infrastructure. The current application does not check the size of the attachment body.

Reason on choosing this topic: We chose this topic as it comes on number 7 in the OWASP Top 10 list. Malicious users often try to send abnormally large amount/size of data which if not handled properly will lead to outage in servers. This will result in downtime which will negatively impact the availability of the software. Thereby reducing its reputation in the market.

Solution:
Installed WAF on the load balancer with a constraint of 50kb on the attachment body. Thus, the user would not be able to upload huge files and the request will be forbidden.

Screenshots:

1. Uploading large file with WAF down



https://csye6225-spring2019-ranadiven.me/note/fbc82479-6429-4540-a00f-920c57bce251/attachments

POST https://csye6225-spring2019-ranadiven.me/note/fbc82479-6429-4540-a00f-920c57bce251/attachments

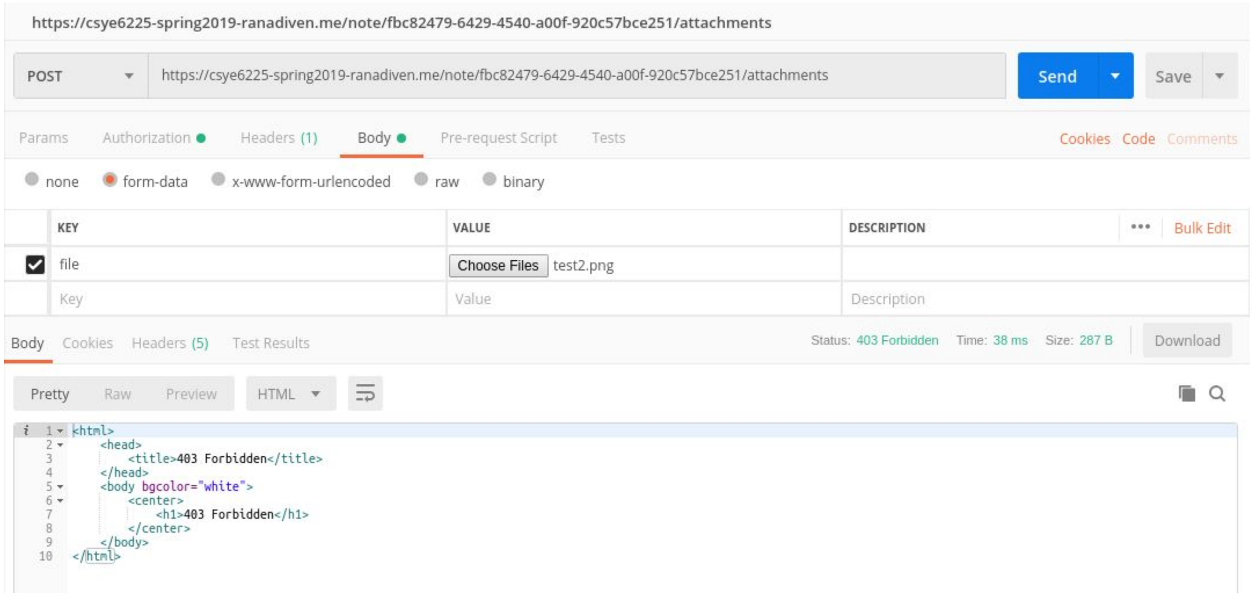
Body: form-data

KEY	VALUE	DESCRIPTION
✓ file	test2.png	

Status: 201 Created Time: 2076 ms Size: 489 B

```
{
    "id": "f39ac9b5-28f2-4bf3-960e-6e740162224c",
    "url": "https://csye6225-spring2019-ranadiven.me.csye6225.com.s3.amazonaws.com/1554390056466-test2.png"
}
```

2. Forbidden request to upload large file when WAF is up



https://csye6225-spring2019-ranadiven.me/note/fbc82479-6429-4540-a00f-920c57bce251/attachments

POST https://csye6225-spring2019-ranadiven.me/note/fbc82479-6429-4540-a00f-920c57bce251/attachments

Body: form-data

KEY	VALUE	DESCRIPTION
✓ file	test2.png	

Status: 403 Forbidden Time: 38 ms Size: 287 B

```
<html>
    <head>
        <title>403 Forbidden</title>
    </head>
    <body bgcolor="white">
        <center>
            <h1>403 Forbidden</h1>
        </center>
    </body>
</html>
```

Case 2: SQL Injection

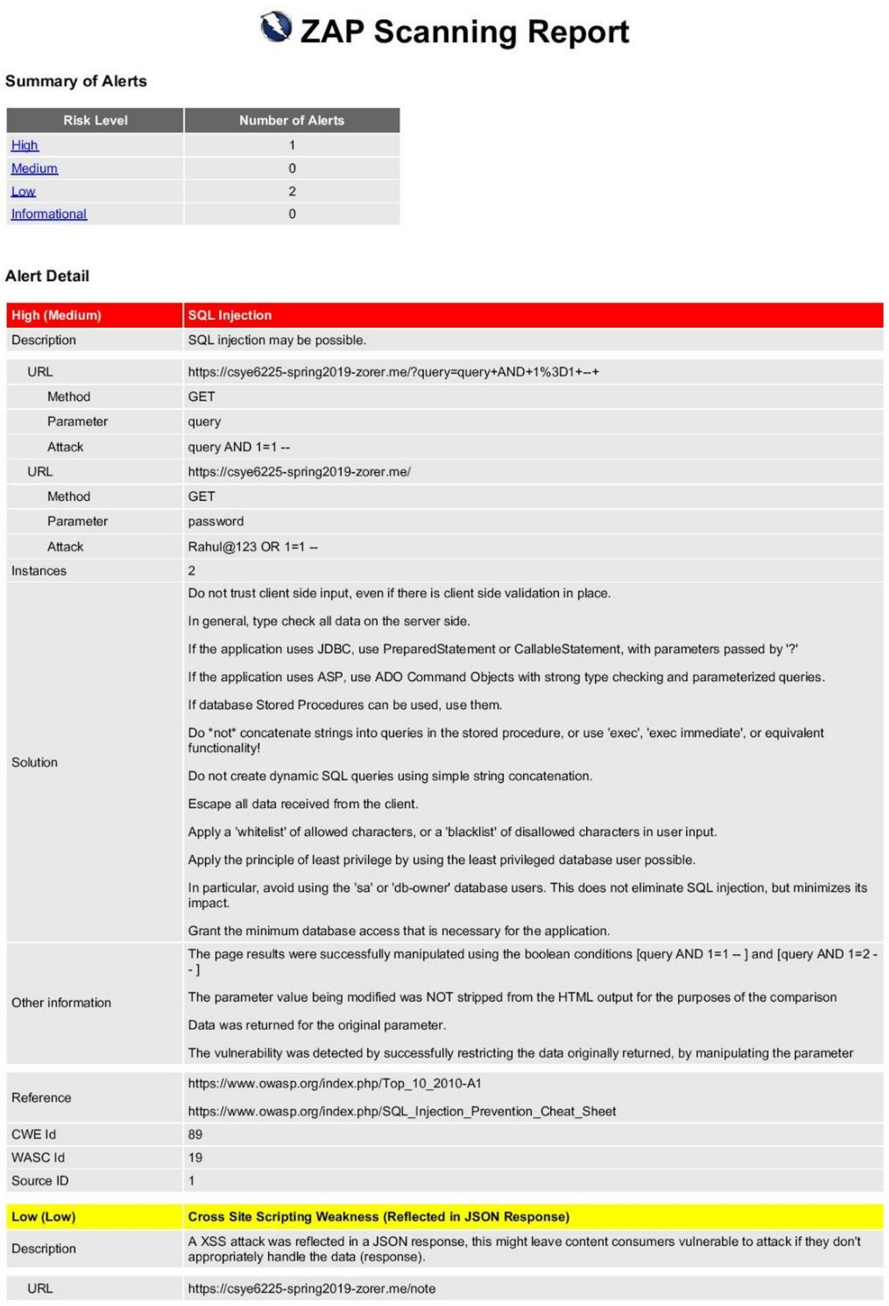
SQL injection is one of the common way attackers try to inject code that executes some script that attacks some security vulnerability in the application. If a SQL injection is successful then the attacker can exploit sensitive data existing in the application database.

Reason on choosing this topic: Sql injection as we all know has been consistently ranked #1 in the OWASP Top 10 list for the past few years , had to be one of our attack vectors for penetration testing. Even after all the new frameworks developed within these years to prevent SQL injection, hackers has still found some or the other way to retrieve data from the database. In addition to these frameworks adding a firewall definitely helps in slowing down the hackers/attackers.

Solution:
With WAF turned on most of the SQL vulnerable requests were blocked successfully.

Screenshots:

1. SQL injection without WAF



ZAP Scanning Report

Summary of Alerts

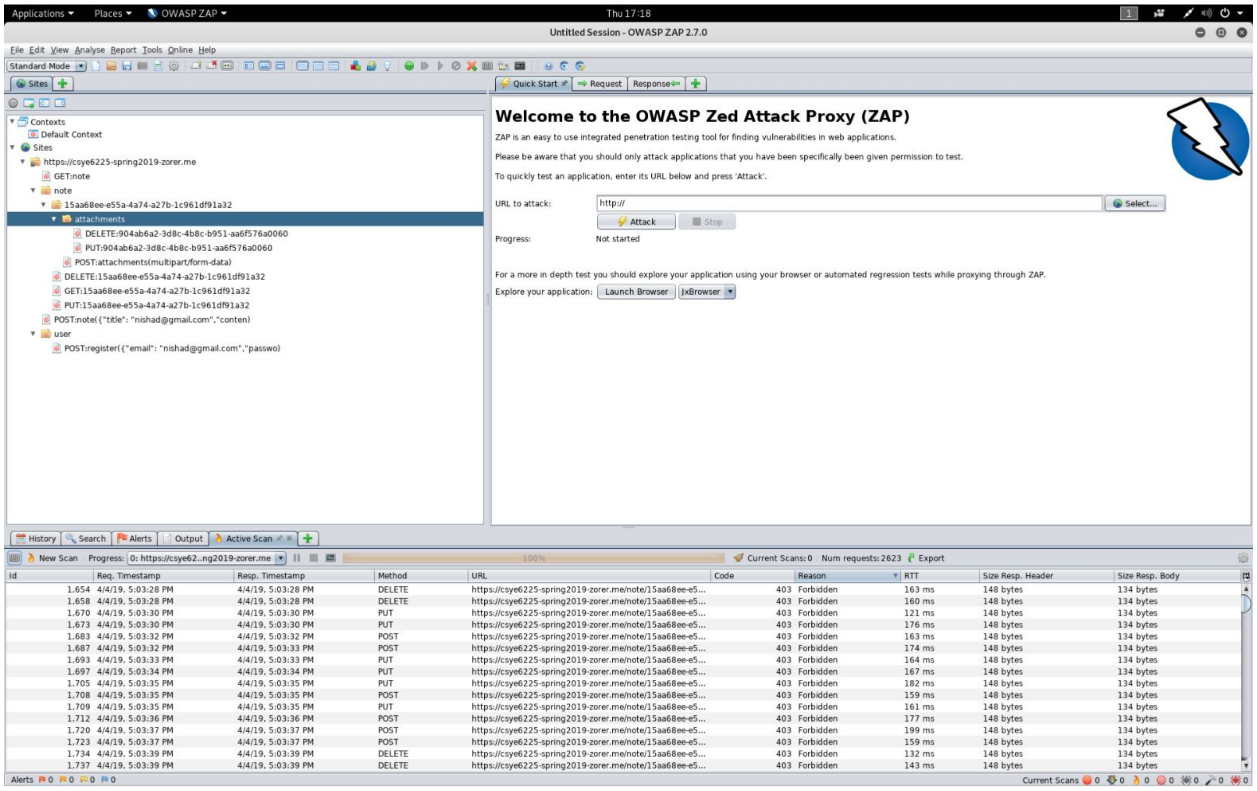
Risk Level	Number of Alerts
High	1
Medium	0
Low	2
Informational	0

Alert Detail

High (Medium)	SQL Injection
Description	SQL injection may be possible.
URL	https://csye6225-spring2019-zorer.me/?query=query+AND+1%3D1+--+
Method	GET
Parameter	query
Attack	query AND 1=1 --
URL	https://csye6225-spring2019-zorer.me/
Method	GET
Parameter	password
Attack	Rahul@123 OR 1=1 --
Instances	2
Solution	Do not trust client side input, even if there is client side validation in place. In general, type check all data on the server side. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?' If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries. If database Stored Procedures can be used, use them. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality! Do not create dynamic SQL queries using simple string concatenation. Escape all data received from the client. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input. Apply the principle of least privilege by using the least privileged database user possible. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact. Grant the minimum database access that is necessary for the application.
Other information	The page results were successfully manipulated using the boolean conditions [query AND 1=1 --] and [query AND 1=2 --] The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter
Reference	https://www.owasp.org/index.php/Top_10_2010-A1 https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
CWE Id	89
WASC Id	19
Source ID	1

Low (Low)	Cross Site Scripting Weakness (Reflected in JSON Response)
Description	A XSS attack was reflected in a JSON response, this might leave content consumers vulnerable to attack if they don't appropriately handle the data (response).
URL	https://csye6225-spring2019-zorer.me/note

2. SQL injection WAF on



Untitled Session - OWASP ZAP 2.7.0

```
▼ Contexts
    Default Context
▼ Sites
  ▼ https://csye6225-spring2019-zorer.me
      GET:note
    ▼ note
      ▼ 15aa68ee-e55a-4a74-a27b-1c961df91a32
        ▼ attachments
            DELETE:904ab6a2-3d8c-4b8c-b951-aa6f576a0060
            PUT:904ab6a2-3d8c-4b8c-b951-aa6f576a0060
          POST:attachments(multipart/form-data)
        DELETE:15aa68ee-e55a-4a74-a27b-1c961df91a32
        GET:15aa68ee-e55a-4a74-a27b-1c961df91a32
        PUT:15aa68ee-e55a-4a74-a27b-1c961df91a32
      POST:note({"title": "nishad@gmail.com","conten)
    ▼ user
        POST:register({"email": "nishad@gmail.com","passwo)
```

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

To quickly test an application, enter its URL below and press 'Attack'.

URL to attack: http://

Progress: Not started

For a more in depth test you should explore your application using your browser or automated regression tests while proxying through ZAP.

Explore your application:

Progress: 0: https://csye62..ng2019-zorer.me 100% Current Scans: 0 Num requests: 2623

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
1,654	4/4/19, 5:03:28 PM	4/4/19, 5:03:28 PM	DELETE	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	163 ms	148 bytes	134 bytes
1,658	4/4/19, 5:03:28 PM	4/4/19, 5:03:28 PM	DELETE	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	160 ms	148 bytes	134 bytes
1,670	4/4/19, 5:03:30 PM	4/4/19, 5:03:30 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	121 ms	148 bytes	134 bytes
1,673	4/4/19, 5:03:30 PM	4/4/19, 5:03:30 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	176 ms	148 bytes	134 bytes
1,683	4/4/19, 5:03:32 PM	4/4/19, 5:03:32 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	163 ms	148 bytes	134 bytes
1,687	4/4/19, 5:03:32 PM	4/4/19, 5:03:33 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	174 ms	148 bytes	134 bytes
1,693	4/4/19, 5:03:33 PM	4/4/19, 5:03:33 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	164 ms	148 bytes	134 bytes
1,697	4/4/19, 5:03:34 PM	4/4/19, 5:03:34 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	167 ms	148 bytes	134 bytes
1,705	4/4/19, 5:03:35 PM	4/4/19, 5:03:35 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	182 ms	148 bytes	134 bytes
1,708	4/4/19, 5:03:35 PM	4/4/19, 5:03:35 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	159 ms	148 bytes	134 bytes
1,709	4/4/19, 5:03:35 PM	4/4/19, 5:03:35 PM	PUT	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	161 ms	148 bytes	134 bytes
1,712	4/4/19, 5:03:36 PM	4/4/19, 5:03:36 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	177 ms	148 bytes	134 bytes
1,720	4/4/19, 5:03:37 PM	4/4/19, 5:03:37 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	199 ms	148 bytes	134 bytes
1,723	4/4/19, 5:03:37 PM	4/4/19, 5:03:37 PM	POST	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	159 ms	148 bytes	134 bytes
1,734	4/4/19, 5:03:39 PM	4/4/19, 5:03:39 PM	DELETE	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	132 ms	148 bytes	134 bytes
1,737	4/4/19, 5:03:39 PM	4/4/19, 5:03:39 PM	DELETE	https://csye6225-spring2019-zorer.me/note/15aa68ee-e5...	403	Forbidden	143 ms	148 bytes	134 bytes

Alerts 0 0 0 0 Current Scans 0 0 0 0 0 0 0

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	0
Low	0
Informational	0

Alert Detail

Case 3: Path traversal LFI, RFI

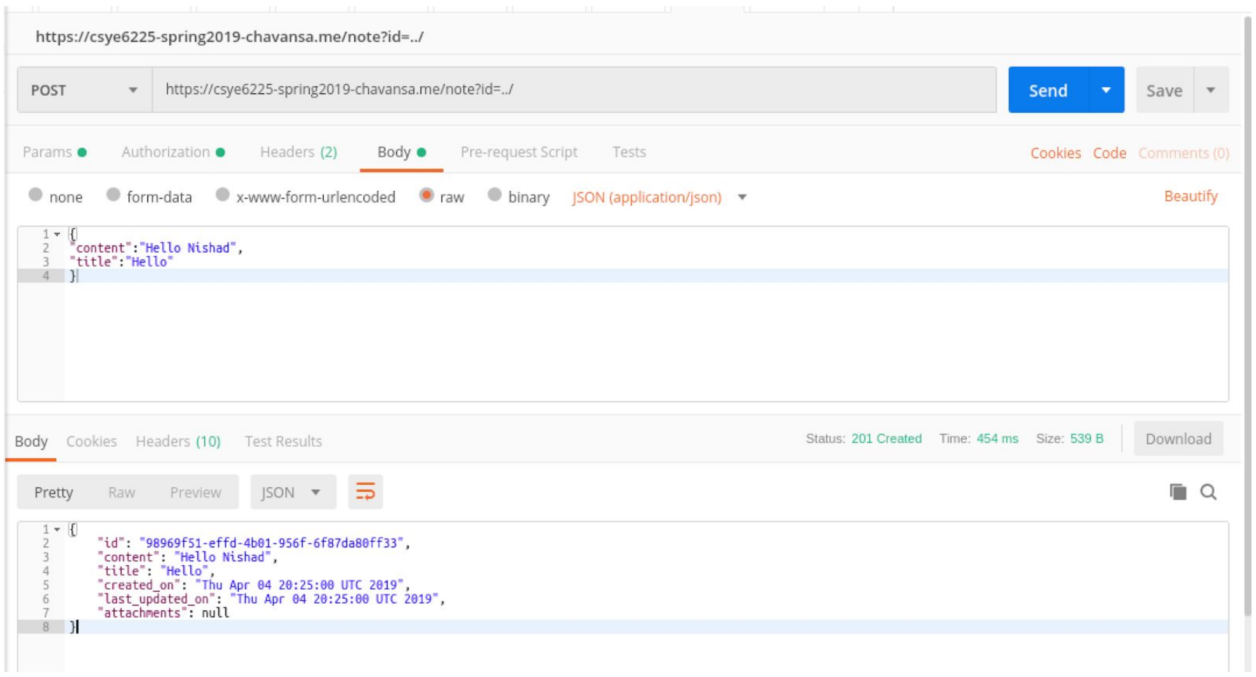
A path traversal attack (also known as directory traversal) tries to access files and directories stored outside the web root folder. It manipulates the variables that reference files with “dot-dot-slash (../)” sequences and its variations or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system including application source code or configuration and critical system files.

Reason on choosing this topic: Directory Traversal ranks #4 in the OWASP top ten list. Hence, it should not be overlooked as it is used by many attackers to get access to files and directories outside web root folder.

Solution: Directory Traversal attacks can be prevented by using AWS WAF by blocking the URI requests and query strings that consist of following pattern “../, ://” after decoding.

Screenshots:

1. Without applying WAF



https://csye6225-spring2019-chavansa.me/note?id=../

POST https://csye6225-spring2019-chavansa.me/note?id=../

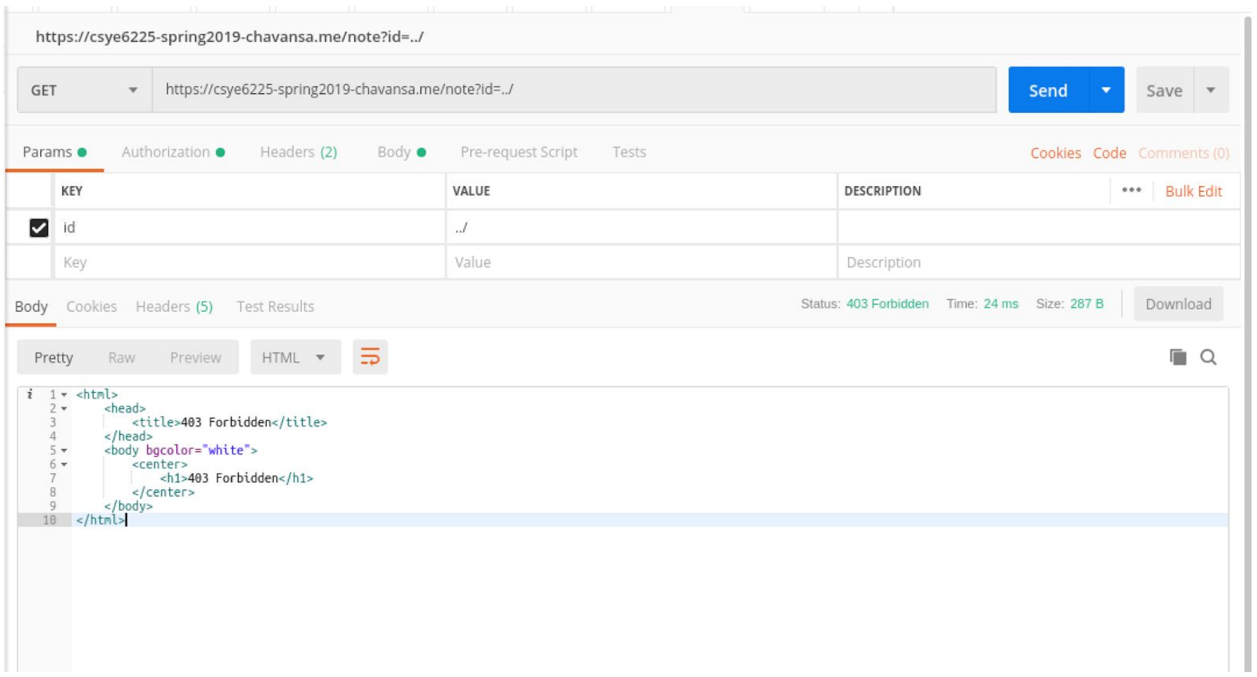
Body: raw, JSON (application/json)

```
{
"content":"Hello Nishad",
"title":"Hello"
}
```

Status: 201 Created Time: 454 ms Size: 539 B

```
{
    "id": "98969f51-effd-4b01-956f-6f87da80ff33",
    "content": "Hello Nishad",
    "title": "Hello",
    "created_on": "Thu Apr 04 20:25:00 UTC 2019",
    "last_updated_on": "Thu Apr 04 20:25:00 UTC 2019",
    "attachments": null
}
```

2. After applying WAF



https://csye6225-spring2019-chavansa.me/note?id=../

KEY	VALUE	DESCRIPTION
id	../	
Key	Value	Description

Status: 403 Forbidden   Time: 24 ms   Size: 287 B

```
<html>
    <head>
        <title>403 Forbidden</title>
    </head>
    <body bgcolor="white">
        <center>
            <h1>403 Forbidden</h1>
        </center>
    </body>
</html>
```

Case 4: Blocking Malicious IP Addresses

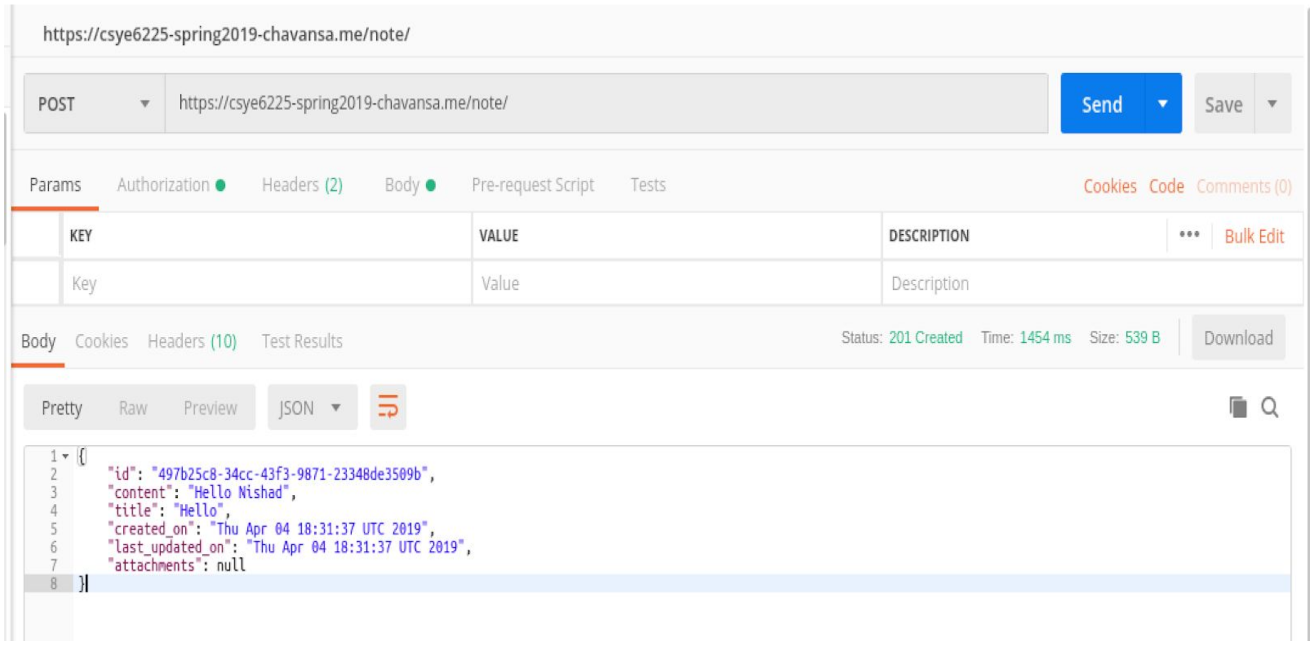
Some IP addresses are known to be used by attackers who are continuously trying to find applications with vulnerabilities and attacking them when they get an opportunity. These IP addresses should be blocked by adding them to a block list and blocking any requests coming from them.

Reason on choosing this topic: Malicious IP addresses ranks #10 in the OWASP Top 10 list but should not be taken for granted. There is data out there which has a list of most of the IP addresses of frequent malicious user or hackers. Blocking such IP addresses definitely helps in slowing down the hackers.

Solution:
The malicious IP addresses can be blocked using AWS WAF by adding these IP addresses or ranges of IP addresses to a block list. Any request coming from these IP addresses should be blocked.

Screenshots:

1. Successful request when IP is not added to block list



https://csye6225-spring2019-chavansa.me/note/

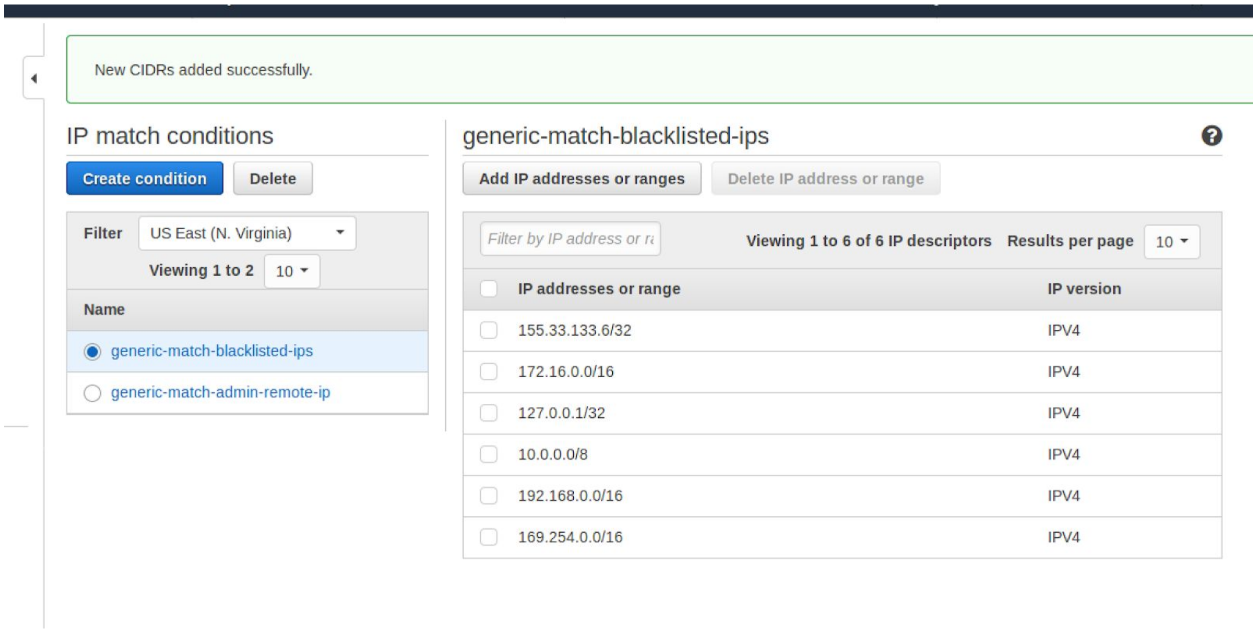
POST https://csye6225-spring2019-chavansa.me/note/

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 201 Created Time: 1454 ms Size: 539 B

```
{
    "id": "497b25c8-34cc-43f3-9871-23348de3509b",
    "content": "Hello Nishad",
    "title": "Hello",
    "created_on": "Thu Apr 04 18:31:37 UTC 2019",
    "last_updated_on": "Thu Apr 04 18:31:37 UTC 2019",
    "attachments": null
}
```

2. Adding my IP address 155.33.133.6/32 to blocklist of WAF



New CIDRs added successfully.

IP match conditions

Create condition
Delete
Filter
US East (N. Virginia)
Viewing 1 to 2
10

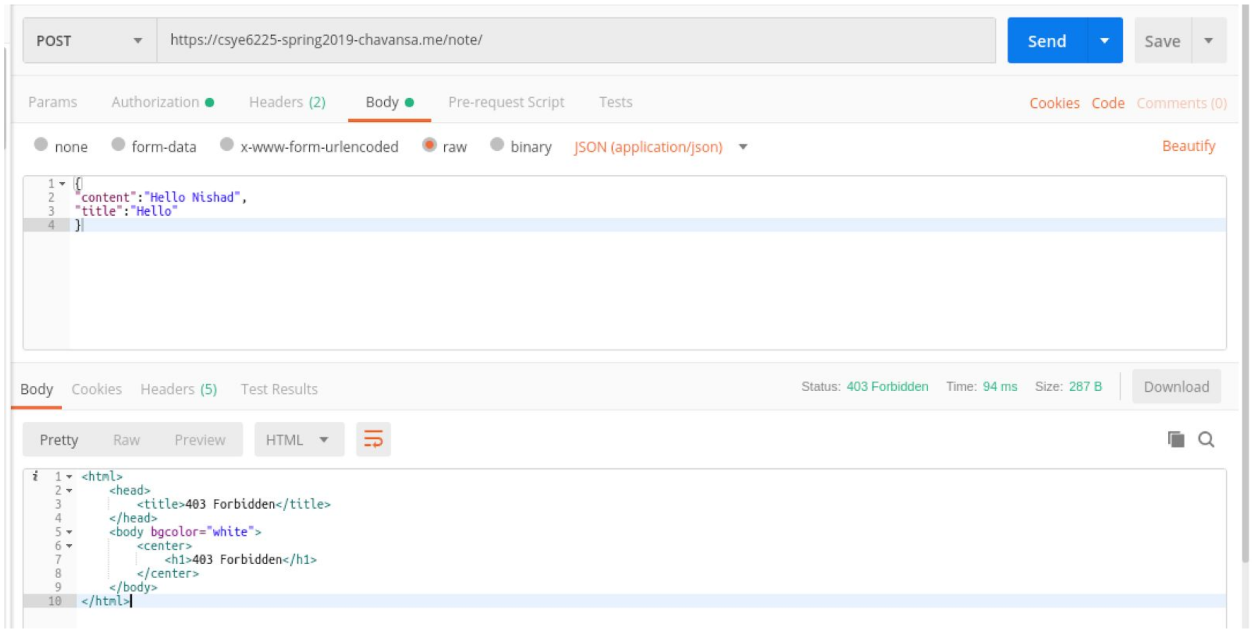
Name
● generic-match-blacklisted-ips
○ generic-match-admin-remote-ip

generic-match-blacklisted-ips

Add IP addresses or ranges
Delete IP address or range
Filter by IP address or ra
Viewing 1 to 6 of 6 IP descriptors
Results per page
10

IP addresses or range	IP version
155.33.133.6/32	IPV4
172.16.0.0/16	IPV4
127.0.0.1/32	IPV4
10.0.0.0/8	IPV4
192.168.0.0/16	IPV4
169.254.0.0/16	IPV4

3. Forbidden request after adding IP address to blocklist.



POST https://csye6225-spring2019-chavansa.me/note/

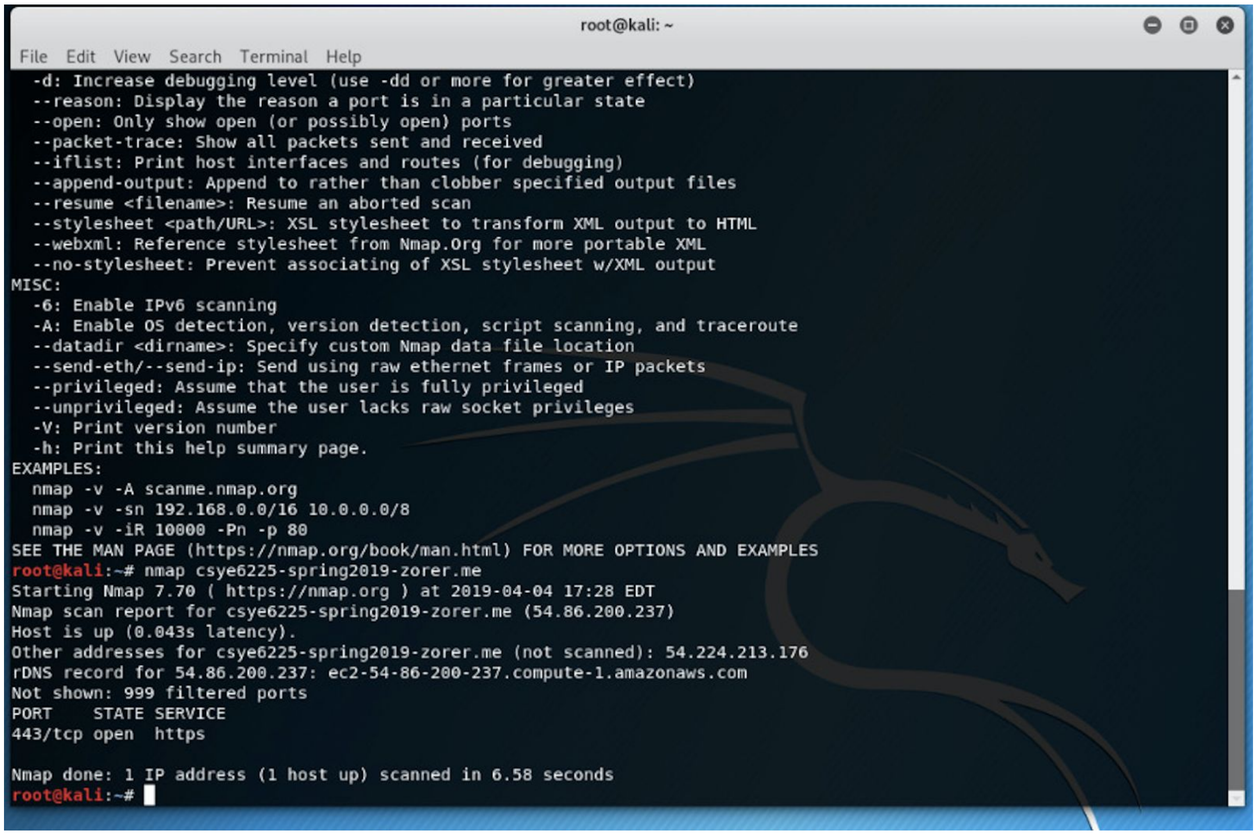
```
{
"content":"Hello Nishad",
"title":"Hello"
}
```

Status: 403 Forbidden Time: 94 ms Size: 287 B

```
<html>
    <head>
        <title>403 Forbidden</title>
    </head>
    <body bgcolor="white">
        <center>
            <h1>403 Forbidden</h1>
        </center>
    </body>
</html>
```

Additional Security Checks:

We have only one secure port open i.e. port 443. We tested this on Kali Linux NMAP Tool and successfully found that only port 443 was open.



```
  -d: Increase debugging level (use -dd or more for greater effect)
  --reason: Display the reason a port is in a particular state
  --open: Only show open (or possibly open) ports
  --packet-trace: Show all packets sent and received
  --iflist: Print host interfaces and routes (for debugging)
  --append-output: Append to rather than clobber specified output files
  --resume <filename>: Resume an aborted scan
  --stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
  --webxml: Reference stylesheet from Nmap.Org for more portable XML
  --no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
  -6: Enable IPv6 scanning
  -A: Enable OS detection, version detection, script scanning, and traceroute
  --datadir <dirname>: Specify custom Nmap data file location
  --send-eth/--send-ip: Send using raw ethernet frames or IP packets
  --privileged: Assume that the user is fully privileged
  --unprivileged: Assume the user lacks raw socket privileges
  -V: Print version number
  -h: Print this help summary page.
EXAMPLES:
  nmap -v -A scanme.nmap.org
  nmap -v -sn 192.168.0.0/16 10.0.0.0/8
  nmap -v -iR 10000 -Pn -p 80
SEE THE MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
root@kali:~# nmap csye6225-spring2019-zorer.me
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-04 17:28 EDT
Nmap scan report for csye6225-spring2019-zorer.me (54.86.200.237)
Host is up (0.043s latency).
Other addresses for csye6225-spring2019-zorer.me (not scanned): 54.224.213.176
rDNS record for 54.86.200.237: ec2-54-86-200-237.compute-1.amazonaws.com
Not shown: 999 filtered ports
PORT    STATE SERVICE
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 6.58 seconds
root@kali:~#
```

NOTE:
We have changed the WAF rule for CSRF to COUNT instead of BLOCK. The reason for this is our webapp has CSRF disabled as of now and enabling CSRF in webapp requires a lot of changes in the code and also in the way we send API requests. But we still have kept the rule as COUNT so that we at least get the number of requests that the WAF was able to identify as CSRF vulnerable requests.

CONCLUSION based on the above findings:
Looking at the above findings, we can conclude that turning on WAF greatly helps in mitigating vulnerabilities especially OWASP Top 10.