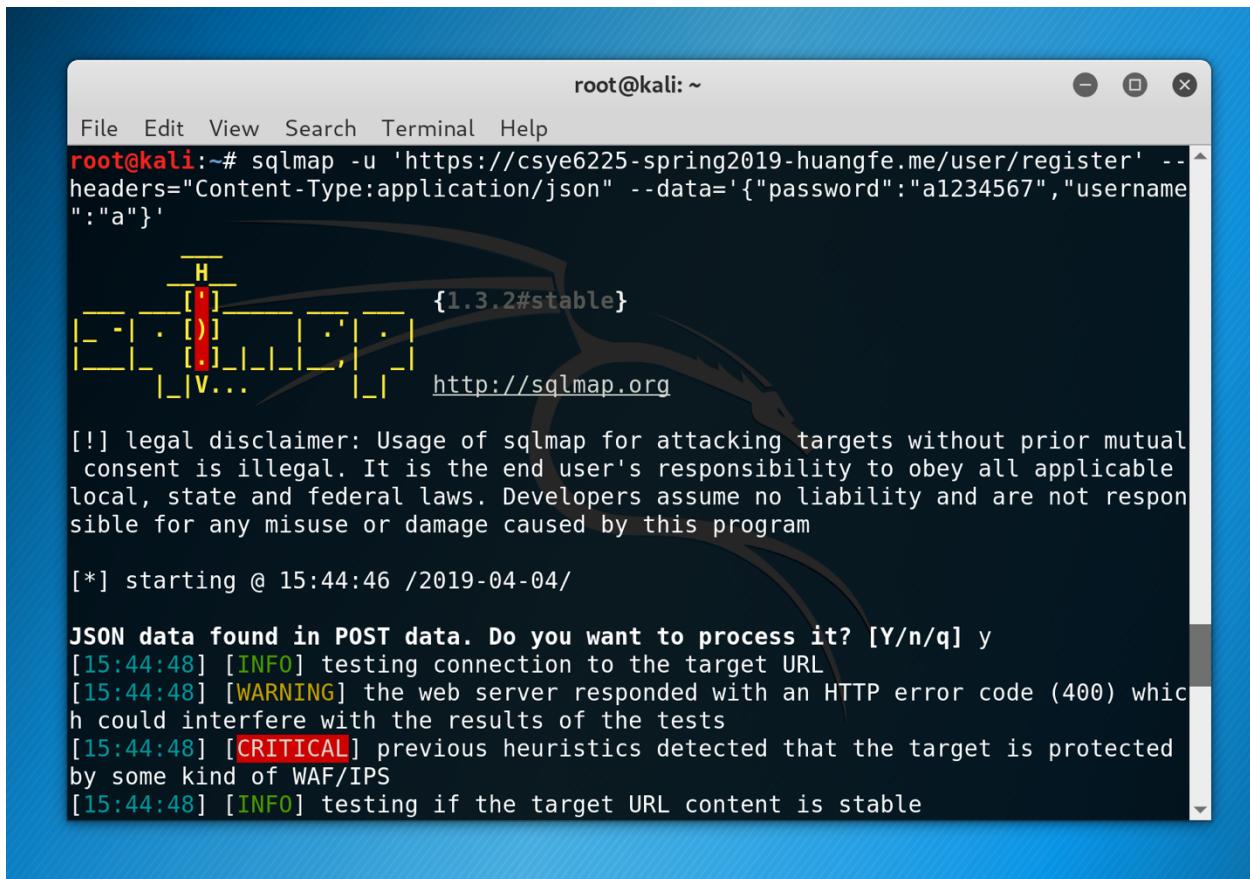


Penetration Testing

1.SQL injection is a code injection technique, used to attack data-driven applications, in which diabolical SQL statements are inserted into an entry field for execution .

We use sqlmap for SQL injection test. For our web application, it checks username in the database when people register user. So, it should be a potential injection point.

We try to inject with or whit out WAF. Whatever the firewall is open, we can resistance attack.



```
root@kali:~# sqlmap -u 'https://csye6225-spring2019-huangfe.me/user/register' --headers="Content-Type:application/json" --data='{"password":"a1234567","username":"a"}'

[!] 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 respon
    sible for any misuse or damage caused by this program

[*] starting @ 15:44:46 /2019-04-04/

JSON data found in POST data. Do you want to process it? [Y/n/q] y
[15:44:48] [INFO] testing connection to the target URL
[15:44:48] [WARNING] the web server responded with an HTTP error code (400) whic
h could interfere with the results of the tests
[15:44:48] [CRITICAL] previous heuristics detected that the target is protected
by some kind of WAF/IPS
[15:44:48] [INFO] testing if the target URL content is stable
```

```

root@kali: ~
File Edit View Search Terminal Help
[15:44:59] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[15:44:59] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[15:44:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[15:44:59] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[15:45:00] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[15:45:00] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[15:45:00] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[15:45:00] [INFO] testing 'Oracle AND time-based blind'
[15:45:01] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[15:45:05] [WARNING] (custom) POST parameter 'JSON username' does not seem to be injectable
[15:45:05] [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') and/or switch '--random-agent'
[15:45:05] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 264 times

[*] ending @ 15:45:05 /2019-04-04/
root@kali:~# 

```

Without WAF, it still can recognize the attack and count it.

Sampled requests

To view new samples, choose [Get new samples](#).

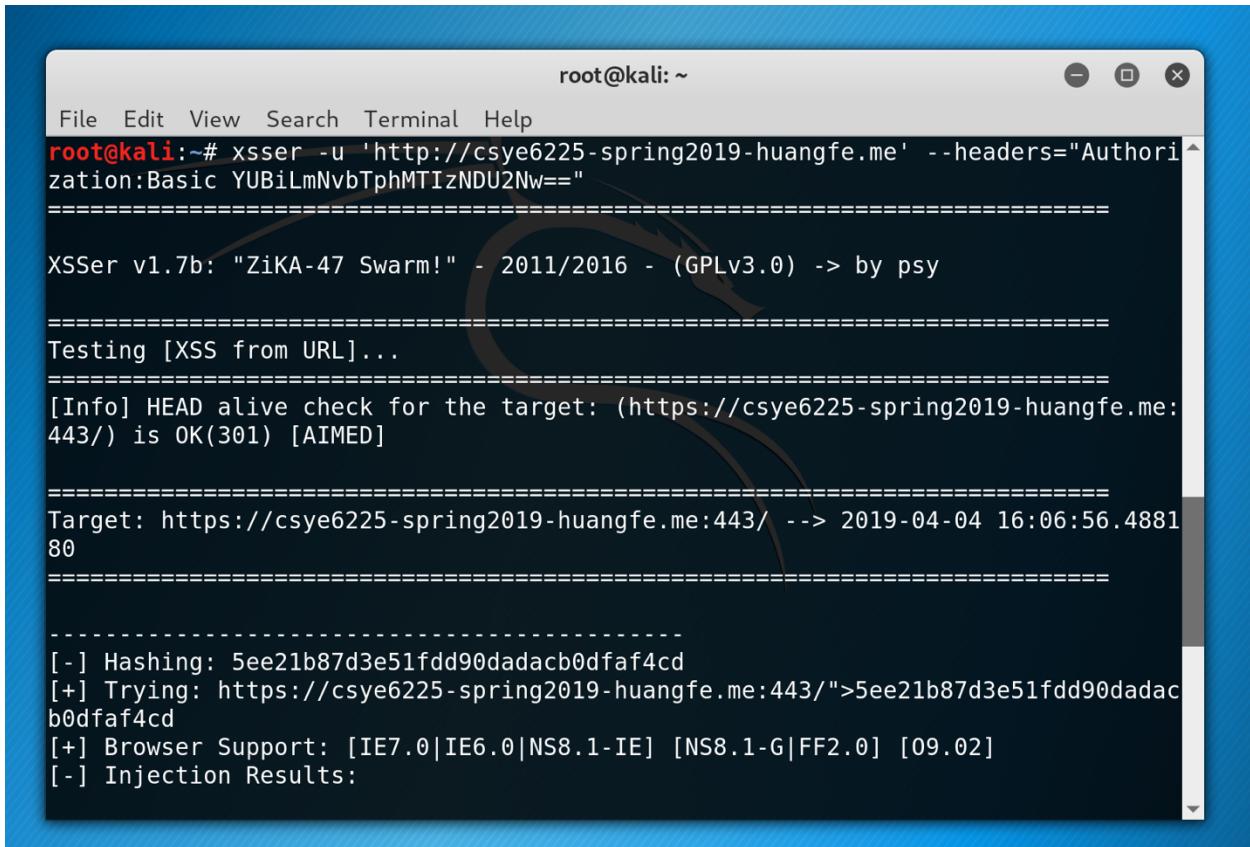
[Get new samples](#)

Sample data from 2019-04-04 19:57:32 to 20:12:32				
Source IP	URI	Matches rule	Action	Time (UTC)
▶ 155.33.134.2	/? Fxjd=6703%20AND%2 01%3D1%20UNION%2 0ALL%20SELECT%201 %2CNULL%2C%27%3 Cscript%3Ealert%28% 22XSS%22%29%3C% 2Fscript%3E%27%2Ct able_name%20FROM% 20information_schema.t ables%20WHERE%202 %3E1-- %2F%2A%2A%2F%3B %20EXEC%20xp_ cmds hell%28%27cat%20.%% 2F.%2F.%2Fetc%2Fpa sswd%27%29%23	mitigate-sqli	Cou nt	20:03:06

We choose this attack because we use MySQL database, and SQL injection is a big threat towards our web application. We must set a string protection for our database.

2. Cross-site scripting (XSS) is a type of injection security attack in which an attacker injects data, such as a malicious script, into content from otherwise trusted websites. Cross-site scripting attacks happen when an untrusted source is allowed to inject its own code into a web application, and that malicious code is included with dynamic content delivered to a victim's browser.

We use xsxer for XSS injection test. We try to inject with or without WAF. Whatever the firewall is open, we can resistance attack.



The screenshot shows a terminal window titled "root@kali: ~". The user has run the command "xsxer -u 'http://csye6225-spring2019-huangfe.me' --headers='Authorization:Basic YUBiLmNvbTphMTIzNDU2Nw=='". The output indicates that the target is alive and OK(301). It also shows the target URL as https://csye6225-spring2019-huangfe.me:443/. The tool then performs hashing, tries to inject, checks browser support (IE7.0|IE6.0|NS8.1-IE, NS8.1-G|FF2.0, 09.02), and finally displays injection results.

```
root@kali:~# xsxer -u 'http://csye6225-spring2019-huangfe.me' --headers="Authorization:Basic YUBiLmNvbTphMTIzNDU2Nw=="
=====
XSSer v1.7b: "ZiKA-47 Swarm!" - 2011/2016 - (GPLv3.0) -> by psy
=====
Testing [XSS from URL]...
=====
[Info] HEAD alive check for the target: (https://csye6225-spring2019-huangfe.me:443/) is OK(301) [AIMED]
=====
Target: https://csye6225-spring2019-huangfe.me:443/ --> 2019-04-04 16:06:56.488180
=====

[-] Hashing: 5ee21b87d3e51fdd90dadacb0dfaf4cd
[+] Trying: https://csye6225-spring2019-huangfe.me:443/">5ee21b87d3e51fdd90dadacb0dfaf4cd
[+] Browser Support: [IE7.0|IE6.0|NS8.1-IE] [NS8.1-G|FF2.0] [09.02]
[-] Injection Results:
```

```
root@kali: ~
File Edit View Search Terminal Help
400 Bad Request: The request could not be understood by the server due to malformed syntax
=====
Mosquito(es) landed!
=====
[*] Final Results:
=====
- Injections: 1
- Failed: 1
- Successful: 0
- Accur: 0 %

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

=====
root@kali:~#
```

By the way, the WAF even block the request when we try to upload a xml file as attachment.

3.Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.

Since we use REST API for our web application, which is stateless, and we don't render actual web pages, in fact there is no need to detect CSRF.

Result: If we enforce CSRF in AWS WAF, any post requests without x-csrf-token header will be blocked. Since we don't check the actual value of x-csrf-token, any value of 36 length will be allowed.

Without WAF:

POST https://csye6225-spring2019-huangfe.me/note/ Send

Params	Authorization	Headers (3)	Body	Pre-request Script	Tests	Cookies	Code
		KEY	VALUE			DESCRIPTION	...
		Authorization	Basic YUBiLmNvbTphMTIzNDU2Nw==				Bulk Edit
<input checked="" type="checkbox"/>		Content-Type	application/json				
<input type="checkbox"/>		x-csrf-token	123456123456123456123456123456123456123456				
		Key	Value			Description	

Body Cookies Headers (4) Test Results Status: 201 Created Time: 213 ms Size: 347 B

Pretty Raw Preview JSON ↗

```

1 { "noteId": "c4265cc1-00a7-4868-ba7c-116e752cbd59",
  "title": "a@b.com",
  "content": "a1234567",
  "created_on": "2019-04-04T20:16:14.76",
  "last_updated_on": "2019-04-04T20:16:14.76",
  "attachments": null
}

```

With WAF & without x-csrf-token header:

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

POST https://csye6225-spring2019-huangfe.me/note/ Send

Params	Authorization	Headers (3)	Body	Pre-request Script	Tests	Cookies	Code
		KEY	VALUE			DESCRIPTION	...
		Authorization	Basic YUBiLmNvbTphMTIzNDU2Nw==				Bulk Edit
<input checked="" type="checkbox"/>		Content-Type	application/json				
<input type="checkbox"/>		x-csrf-token	123456123456123456123456123456123456				
		Key	Value			Description	

Body Cookies Headers (5) Test Results Status: 403 Forbidden Time: 18 ms Size: 287 B

Pretty Raw Preview HTML ↗

```

1 <html>
2   <head>
3     <title>403 Forbidden</title>
4   </head>
5   <body bgcolor="white">
6     <center>
7       <h1>403 Forbidden</h1>
8     </center>
9   </body>
10 </html>

```

With WAF & with x-csrf-token header:

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

POST https://csye6225-spring2019-huangfe.me/note/ Send

Params Authorization Headers (3) Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Authorization	Basic YUBiLmNvbTphMTIzNDU2Nw==			
<input checked="" type="checkbox"/> Content-Type	application/json			
<input checked="" type="checkbox"/> x-csrf-token	123456123456123456123456123456123456123456			
Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 201 Created Time: 223 ms Size: 349 B

Pretty Raw Preview JSON

```
1 [ {  
2   "noteId": "72b509b7-163b-4c32-ab62-2cddf4fcb5c4",  
3   "title": "a@b.com",  
4   "content": "a1234567",  
5   "created_on": "2019-04-04T20:20:29.696",  
6   "last_updated_on": "2019-04-04T20:20:29.696",  
7   "attachments": null  
8 }
```

Why do we choose this attack?

For most of web applications, it is developer's responsibility to consider all kinds of attack, and CSRF is one of the most common attacks to think about. Therefore, we select this attack.