CSYE 6225 Fall 2019

Web Application Firewall Penetration Testing

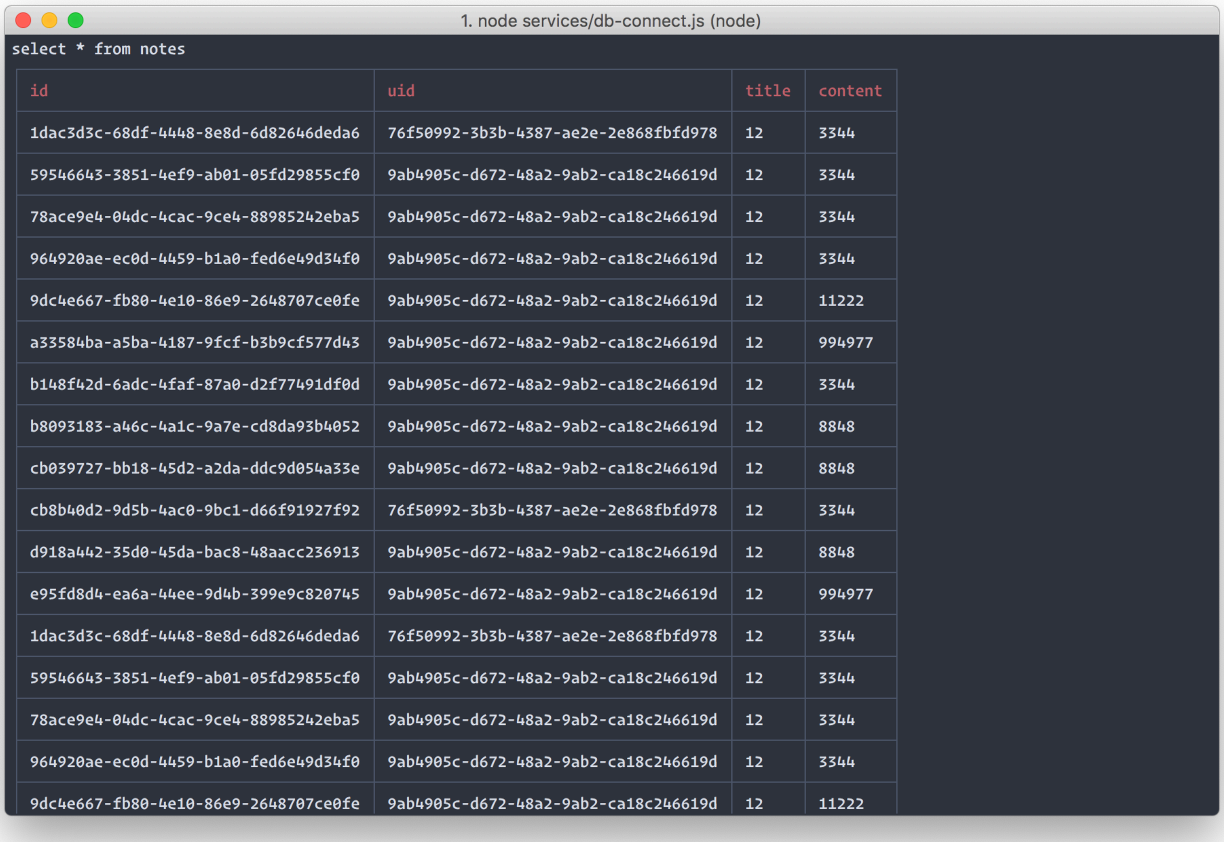
Zhichao Pan(001493794)
Mengying Wang(001357559)
Shiyu Wang(001400142)

1. OWASP Top 10 A1 SQL Injection

Attack Vector:

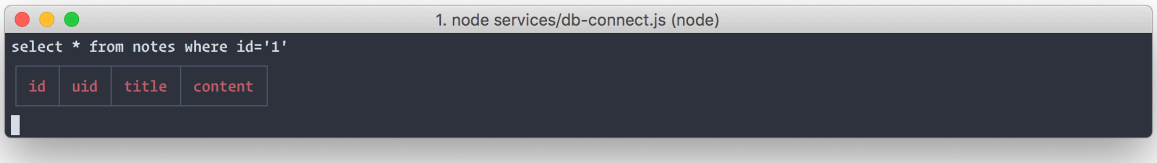
By inserting unsanitary data into requests to the interpreter, an attacker can alter the intent of the requests and cause unexpected actions.

Select * from notes



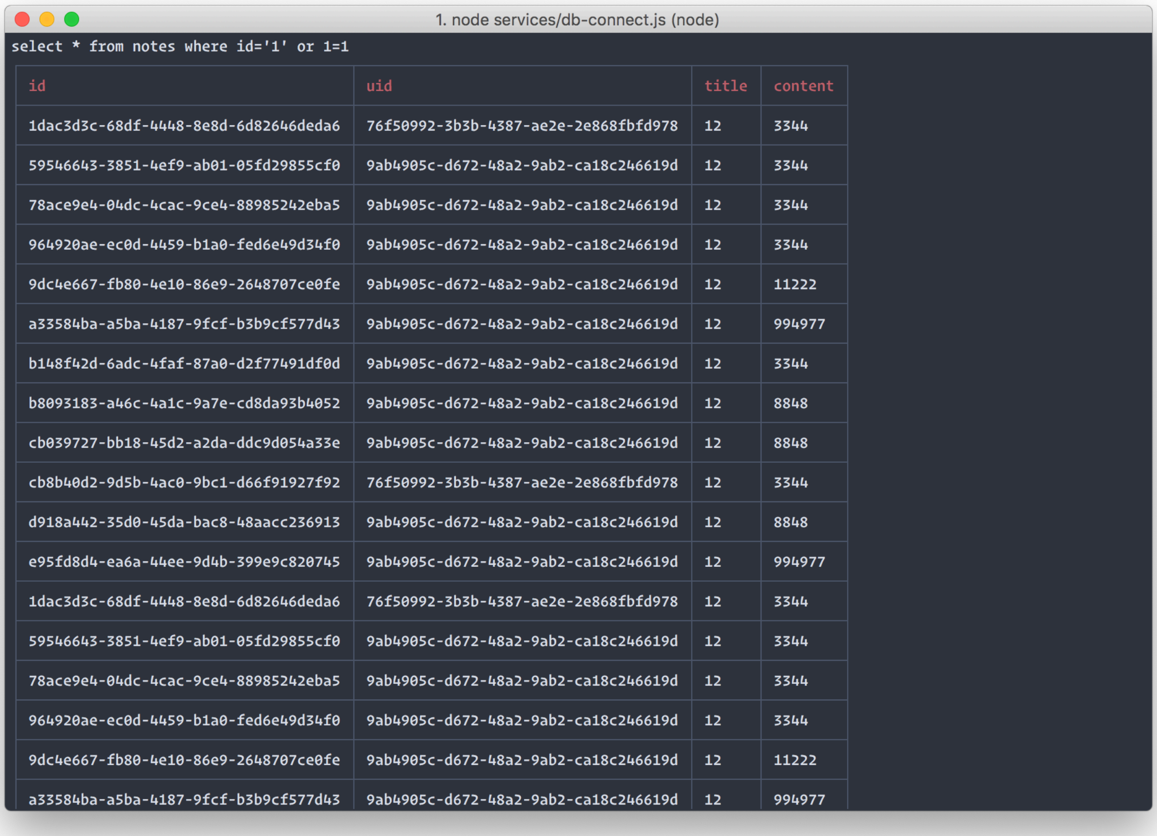
```
select * from notes
| id                                   | uid                                  | title | content |
| 1dac3d3c-68df-4448-8e8d-6d82646deda6 | 76f50992-3b3b-4387-ae2e-2e868fbfd978 | 12    | 3344    |
| 59546643-3851-4ef9-ab01-05fd29855cf0 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 78ace9e4-04dc-4cac-9ce4-88985242eba5 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 964920ae-ec0d-4459-b1a0-fed6e49d34f0 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 9dc4e667-fb80-4e10-86e9-2648707ce0fe | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 11222   |
| a33584ba-a5ba-4187-9fcf-b3b9cf577d43 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 994977  |
| b148f42d-6adc-4faf-87a0-d2f77491df0d | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| b8093183-a46c-4a1c-9a7e-cd8da93b4052 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 8848    |
| cb039727-bb18-45d2-a2da-ddc9d054a33e | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 8848    |
| cb8b40d2-9d5b-4ac0-9bc1-d66f91927f92 | 76f50992-3b3b-4387-ae2e-2e868fbfd978 | 12    | 3344    |
| d918a442-35d0-45da-bac8-48aacc236913 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 8848    |
| e95fd8d4-ea6a-44ee-9d4b-399e9c820745 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 994977  |
| 1dac3d3c-68df-4448-8e8d-6d82646deda6 | 76f50992-3b3b-4387-ae2e-2e868fbfd978 | 12    | 3344    |
| 59546643-3851-4ef9-ab01-05fd29855cf0 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 78ace9e4-04dc-4cac-9ce4-88985242eba5 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 964920ae-ec0d-4459-b1a0-fed6e49d34f0 | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 3344    |
| 9dc4e667-fb80-4e10-86e9-2648707ce0fe | 9ab4905c-d672-48a2-9ab2-ca18c246619d | 12    | 11222   |
```

Select * from notes where id='1'



```
select * from notes where id='1'
| id | uid | title | content |
```

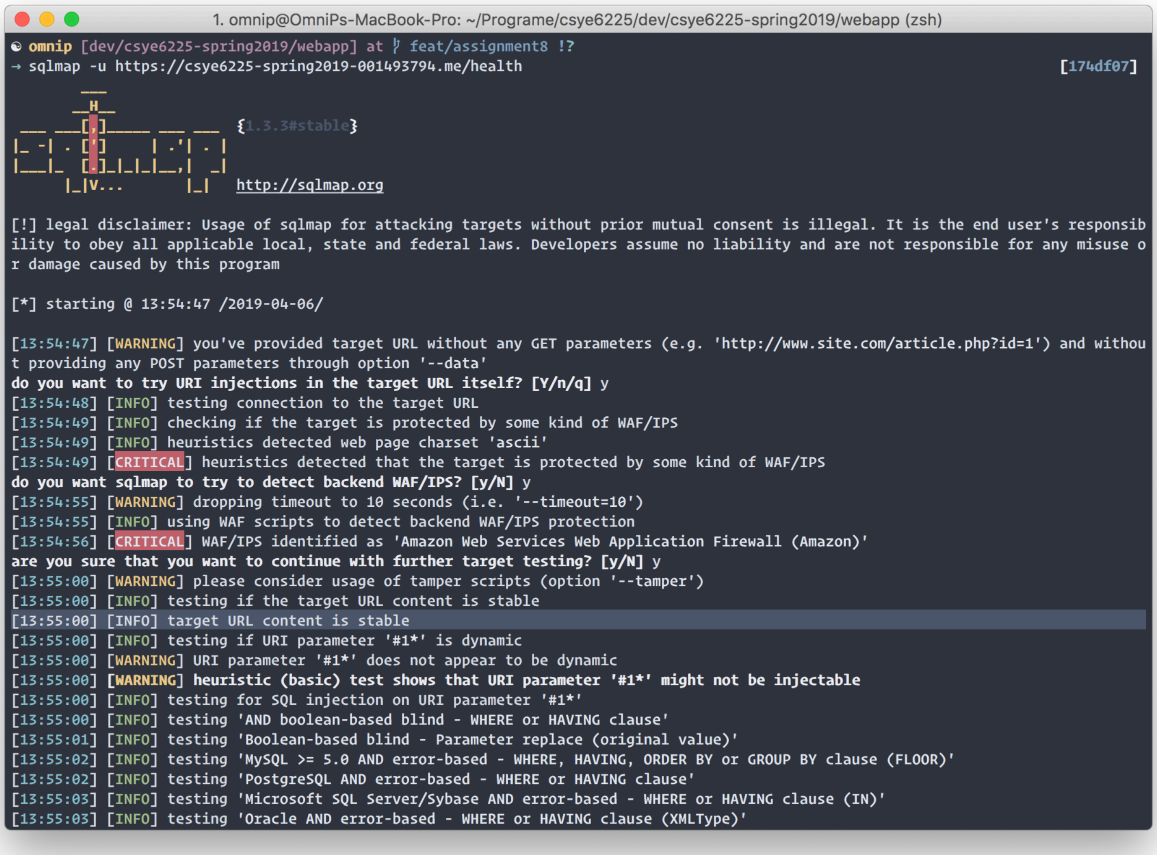
Select * from notes where id='1' or 1=1



```
1. node services/db-connect.js (node)
select * from notes where id='1' or 1=1
 id                                     uid                                    title   content
 1dac3d3c-68df-4448-8e8d-6d82646deda6   76f50992-3b3b-4387-ae2e-2e868fbfd978   12      3344
 59546643-3851-4ef9-ab01-05fd29855cf0   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 78ace9e4-04dc-4cac-9ce4-88985242eba5   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 964920ae-ec0d-4459-b1a0-fed6e49d34f0   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 9dc4e667-fb80-4e10-86e9-2648707ce0fe   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      11222
 a33584ba-a5ba-4187-9fcf-b3b9cf577d43   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      994977
 b148f42d-6adc-4faf-87a0-d2f77491df0d   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 b8093183-a46c-4a1c-9a7e-cd8da93b4052   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      8848
 cb039727-bb18-45d2-a2da-ddc9d054a33e   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      8848
 cb8b40d2-9d5b-4ac0-9bc1-d66f91927f92   76f50992-3b3b-4387-ae2e-2e868fbfd978   12      3344
 d918a442-35d0-45da-bac8-48aacc236913   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      8848
 e95fd8d4-ea6a-44ee-9d4b-399e9c820745   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      994977
 1dac3d3c-68df-4448-8e8d-6d82646deda6   76f50992-3b3b-4387-ae2e-2e868fbfd978   12      3344
 59546643-3851-4ef9-ab01-05fd29855cf0   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 78ace9e4-04dc-4cac-9ce4-88985242eba5   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 964920ae-ec0d-4459-b1a0-fed6e49d34f0   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      3344
 9dc4e667-fb80-4e10-86e9-2648707ce0fe   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      11222
 a33584ba-a5ba-4187-9fcf-b3b9cf577d43   9ab4905c-d672-48a2-9ab2-ca18c246619d   12      994977
```

Result:

After creating WAF, we can see that the target domain is stable on sql map.



```
1. omnip@OmniPs-MacBook-Pro: ~/Programe/csye6225/dev/csye6225-spring2019/webapp (zsh)
omnip [dev/csye6225-spring2019/webapp] at  feat/assignment8 !?
→ sqlmap -u https://csye6225-spring2019-001493794.me/health                                 [174df07]
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.3#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 responsib
ility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse o
r damage caused by this program

[*] starting @ 13:54:47 /2019-04-06/

[13:54:47] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/article.php?id=1') and withou
t providing any POST parameters through option '--data'
do you want to try URI injections in the target URL itself? [Y/n/q] y
[13:54:48] [INFO] testing connection to the target URL
[13:54:49] [INFO] checking if the target is protected by some kind of WAF/IPS
[13:54:49] [INFO] heuristics detected web page charset 'ascii'
[13:54:49] [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
[13:54:55] [WARNING] dropping timeout to 10 seconds (i.e. '--timeout=10')
[13:54:55] [INFO] using WAF scripts to detect backend WAF/IPS protection
[13:54:56] [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
[13:55:00] [WARNING] please consider usage of tamper scripts (option '--tamper')
[13:55:00] [INFO] testing if the target URL content is stable
[13:55:00] [INFO] target URL content is stable
[13:55:00] [INFO] testing if URI parameter '#1*' is dynamic
[13:55:00] [WARNING] URI parameter '#1*' does not appear to be dynamic
[13:55:00] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might not be injectable
[13:55:00] [INFO] testing for SQL injection on URI parameter '#1*'
[13:55:00] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[13:55:01] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[13:55:02] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[13:55:02] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[13:55:03] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[13:55:03] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
```

Why choose this attack vector?

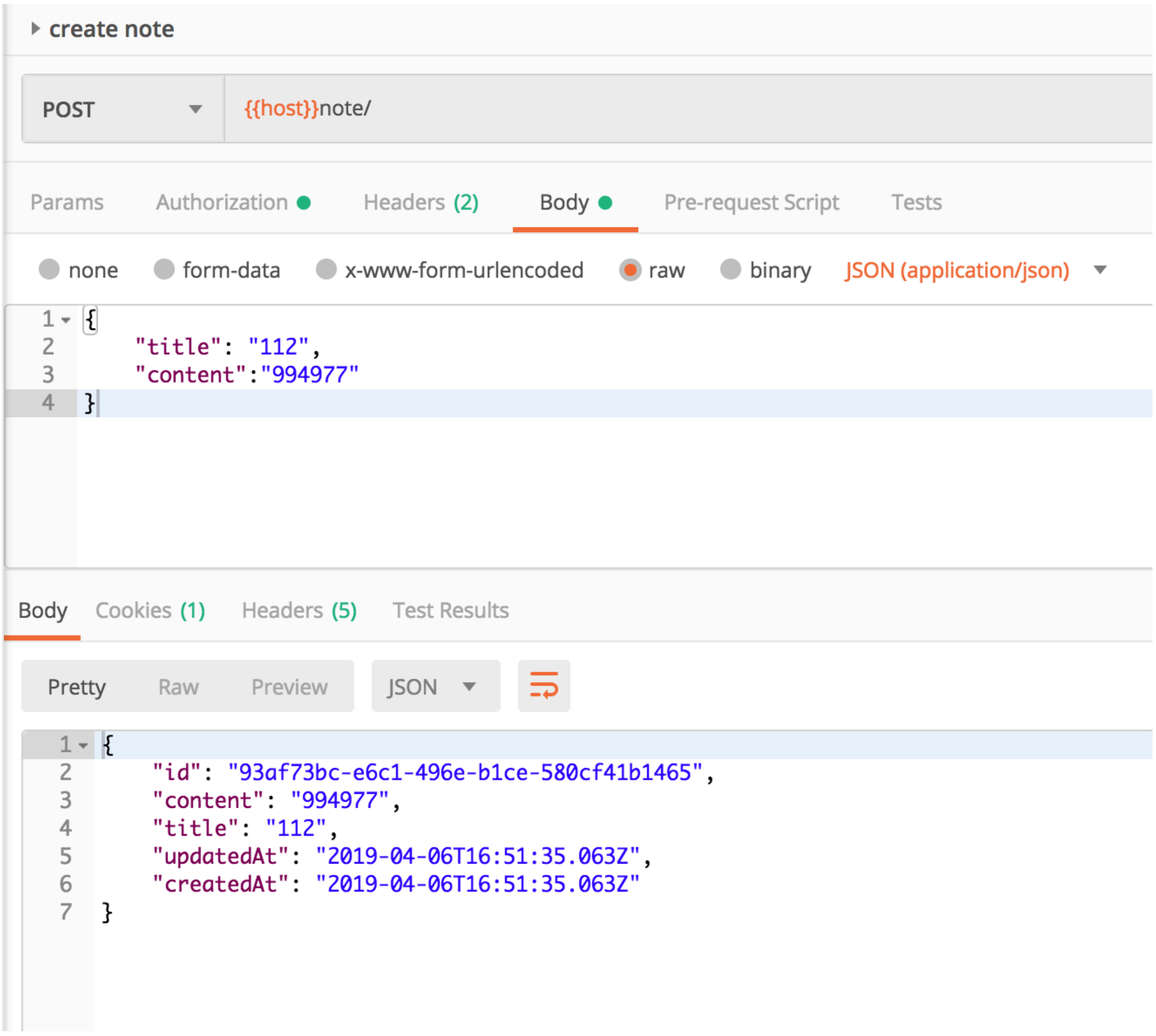
Attackers use almost any source of data as an injection vector, environment variables, parameters, external and internal web services, and all types of users and these vulnerabilities are very often in SQL. Injection can result in data loss or denial of access

2. OWASP Top 10 A8 CSRF

Matching the specific http method and does not match the token size constraint will not be able to access content

Attack Vector:
Hackers or crackers can gain the data which is not belong to them using CSRF. Such as add a url in a phishing site, then trick the user into executing actions. We can resolve it by checking the http method and if the token existed.

Result:
Below access content directly through ip which does not contain CSRF check. As result, the response is success.



POST {{host}}note/

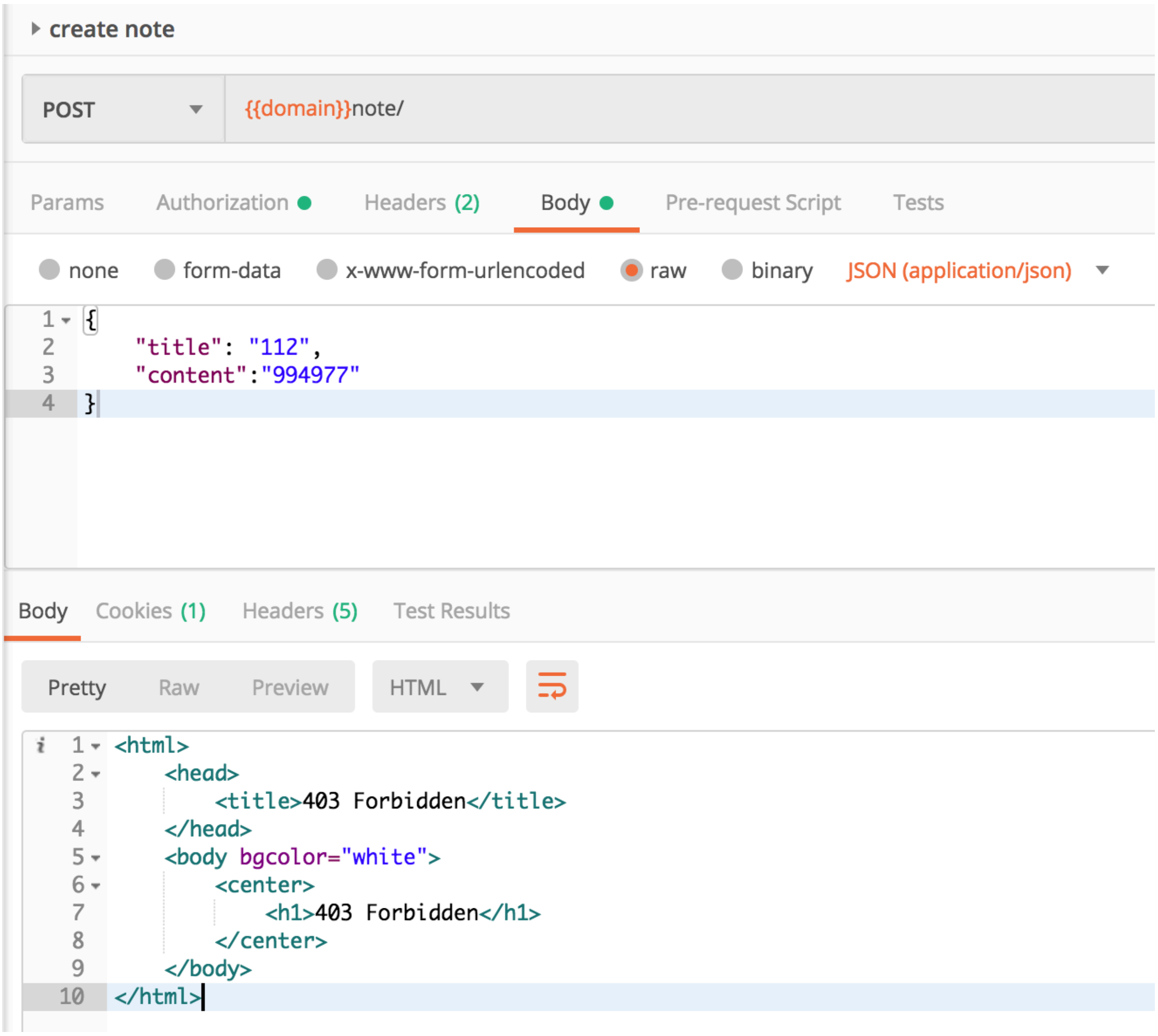
Body: raw, JSON (application/json)

```
{
    "title": "112",
    "content":"994977"
}
```

Response Body (JSON):

```
{
    "id": "93af73bc-e6c1-496e-b1ce-580cf41b1465",
    "content": "994977",
    "title": "112",
    "updatedAt": "2019-04-06T16:51:35.063Z",
    "createdAt": "2019-04-06T16:51:35.063Z"
}
```

This picture is after adopting CSRF check. As it shows, the response status code is 403.



POST {{domain}}note/

Body: raw, JSON (application/json)

```
{
    "title": "112",
    "content":"994977"
}
```

Body: HTML

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

Why choose this vector?

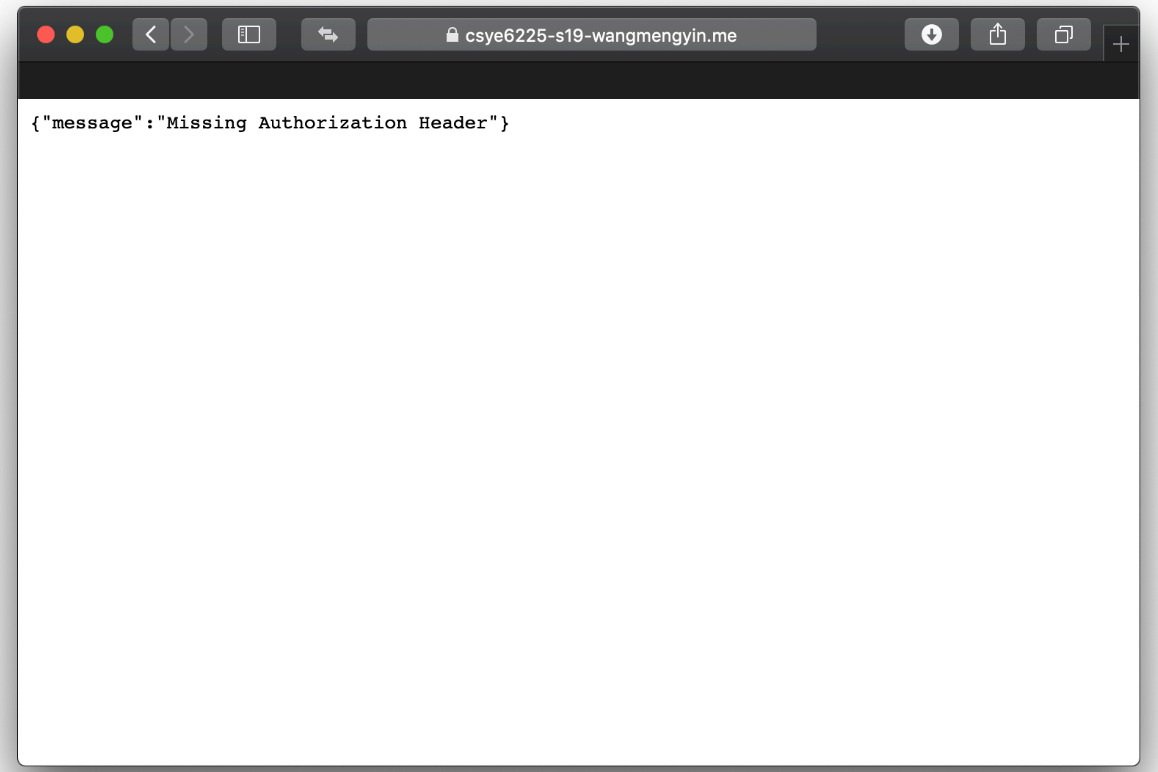
Attackers choose this attack vector as they could access unauthorized data. Which might result in massive lost. Especial some fields related to finance.

3. IP Blacklist

IP addresses in the black list should not be allowed to access content.

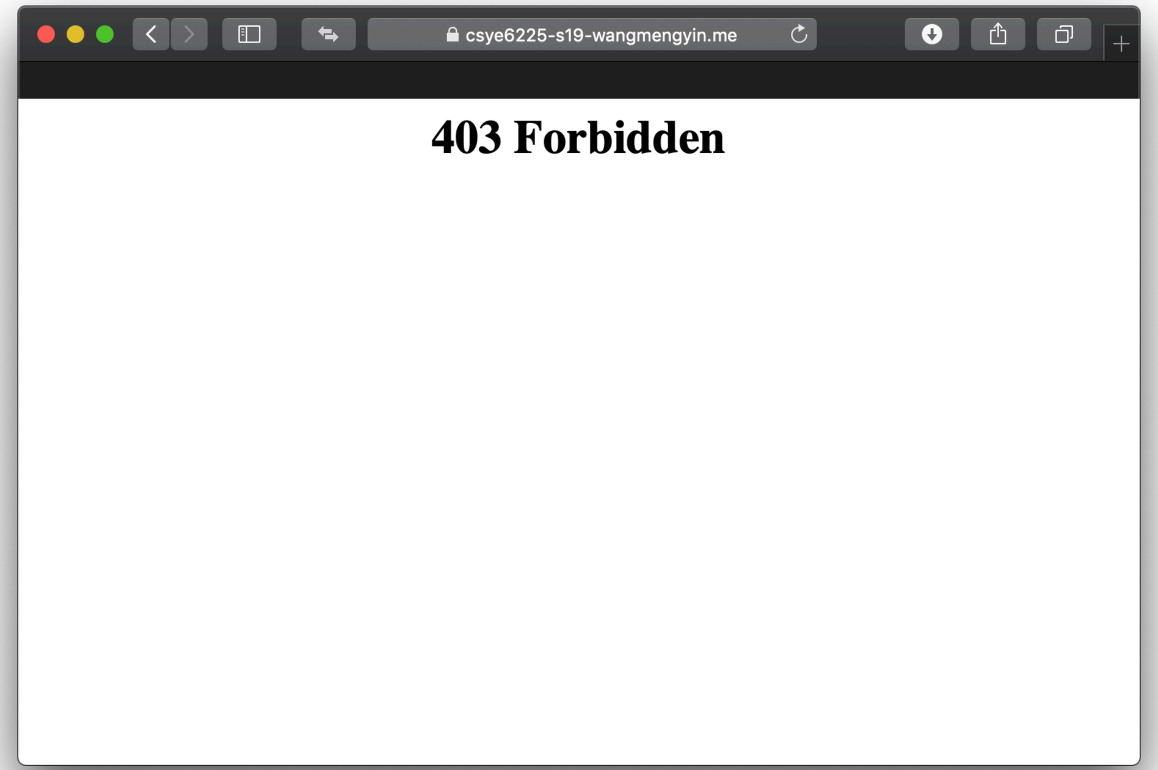
Attack Vector:
Using dangerous IP address to access our domain.
(We can assume that the IP Address I used in this section's test is dangerous, and we need to prevent it to access my domain.)

Result:
As you can see in the next picture, when the WAF was not built, the IP address with risks could access my domain, so the IP address can not be detected and prevented without WAF.



```
{"message":"Missing Authorization Header"}
```

After I built WAF and blacklist the dangerous IP address, it was been stopped when I tried to access my domain via this IP.



403 Forbidden

Why choose this vector?

Attacks from dangerous IP address is very common in our real life, and setting blacklist is an easy and useful way to prevent it. So using this vector to check whether our blacklist is work well is very important for our domain’s safety.