Penetration Testing

1.SQL Injection

Attack Vector: SQL Injection (tool: sqlmap in Kali Linux)

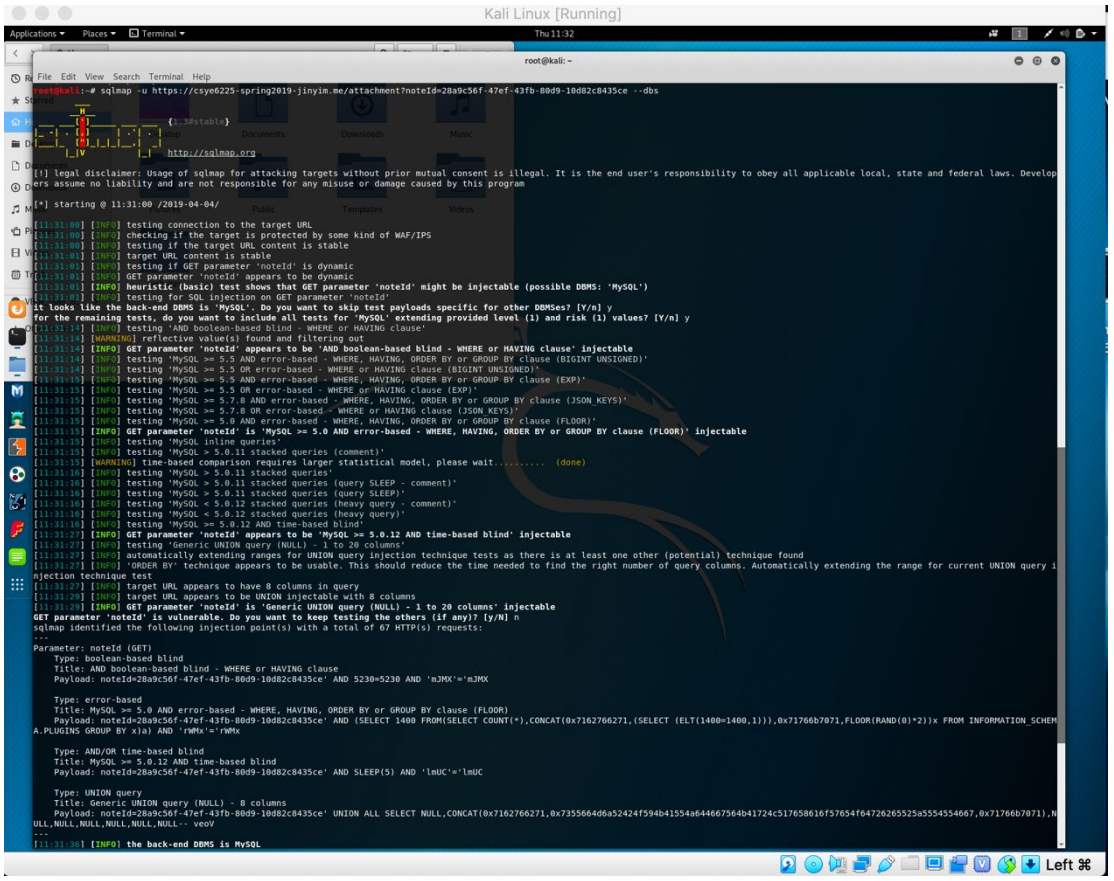
The reason why I choose this Attack Vector:

SQL Injection is A1 in Top 10 OWASP and it is mostly known as an attack vector for websites but can be used to attack any type of SQL database. 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 (e.g. to dump the database contents to the attacker) SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. I choose this as an attack vector because nowadays a lot of people work on database but they don't really care about the sql so they cause a lot of dangerous to their data. Use sql injection can get user's password and some critical information which is very bad for both user and company!!

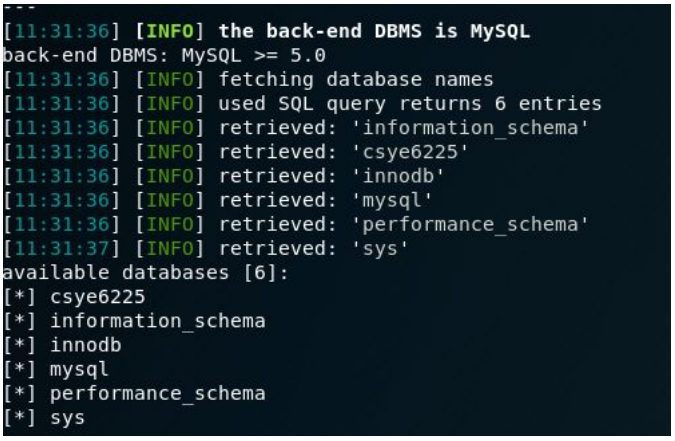
Attack Result:

Before we active aws waf:

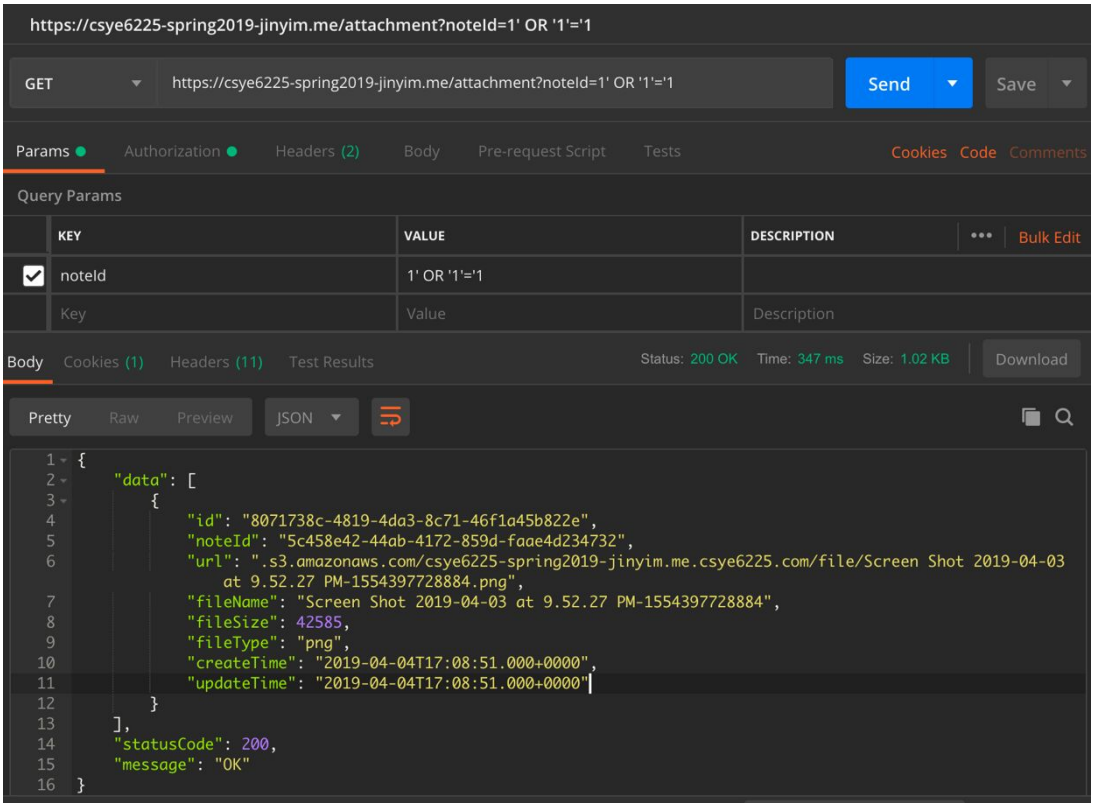
I use sqlmap in Kali Linux to attack one of my web url :https://csye6225-spring2019-jinyim.me/attachment?noteId=5c458e42-44ab-4172-859d-faae4d234732**.** The sqlmap shows that it is **injectable** as you can seen from the picture.



It also shows the detail information about my database:

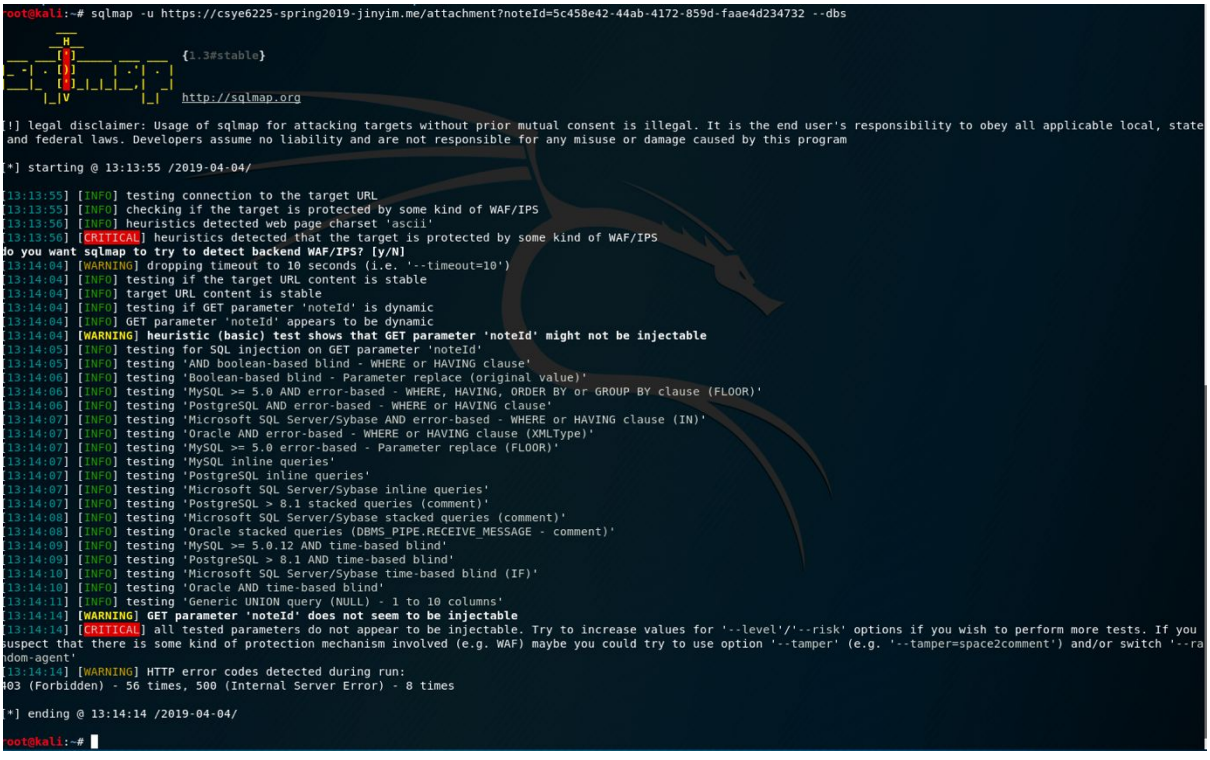


I also use postman to inject a sql with a GET method to test it with “?noteId=1' OR '1'='1”. It turns out I can get all the attachments from different user and this means the database has been Inject successful. The reason is that our select sql is : **select * from attachment where noteId= ‘${noteId}**. If you pass “1’ OR ‘1’ = 1” to {noteID}, the whole sentence will become : **select * from attachment**. Take a look from these pictures.

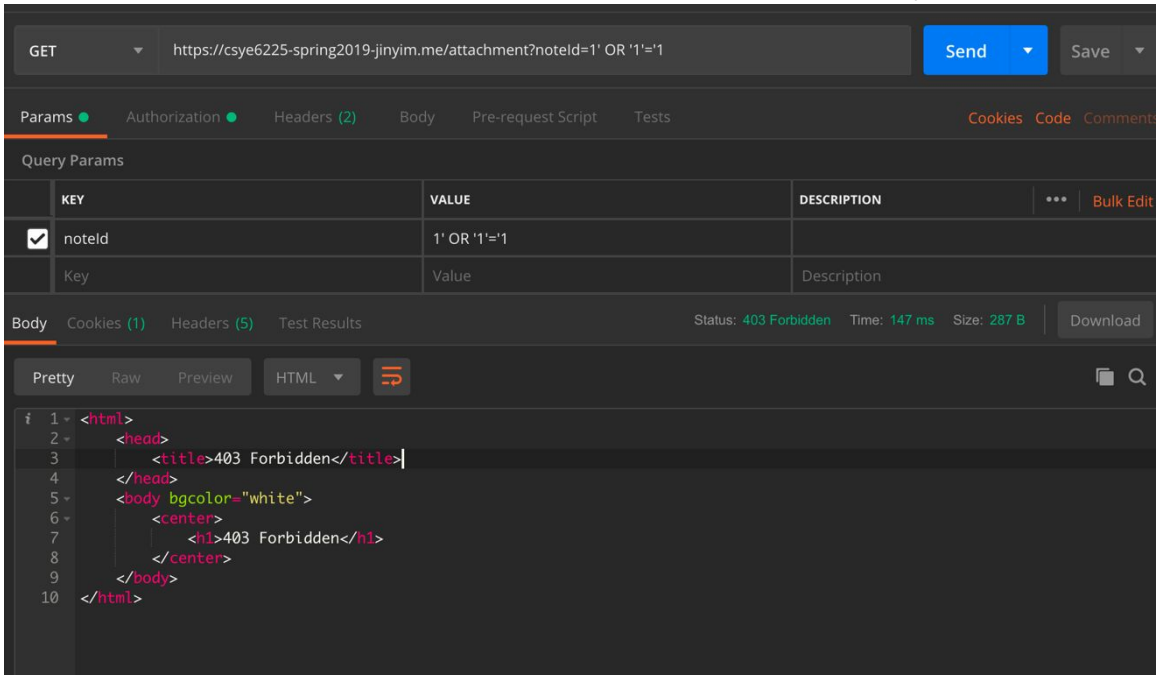


After we active aws waf:

all these sql Injection will be useless. Let's look at the sqlmap result:



As you can see that heuristics detected that the target is protected by some kind of WAF/IPS, it also shows that GET parameters does not seem to be injectable. The same result from the postman: I use the exactly same sql Injection but it give me a 403 Forbidden. Both result shows that waf work very well.



2.XSS attack

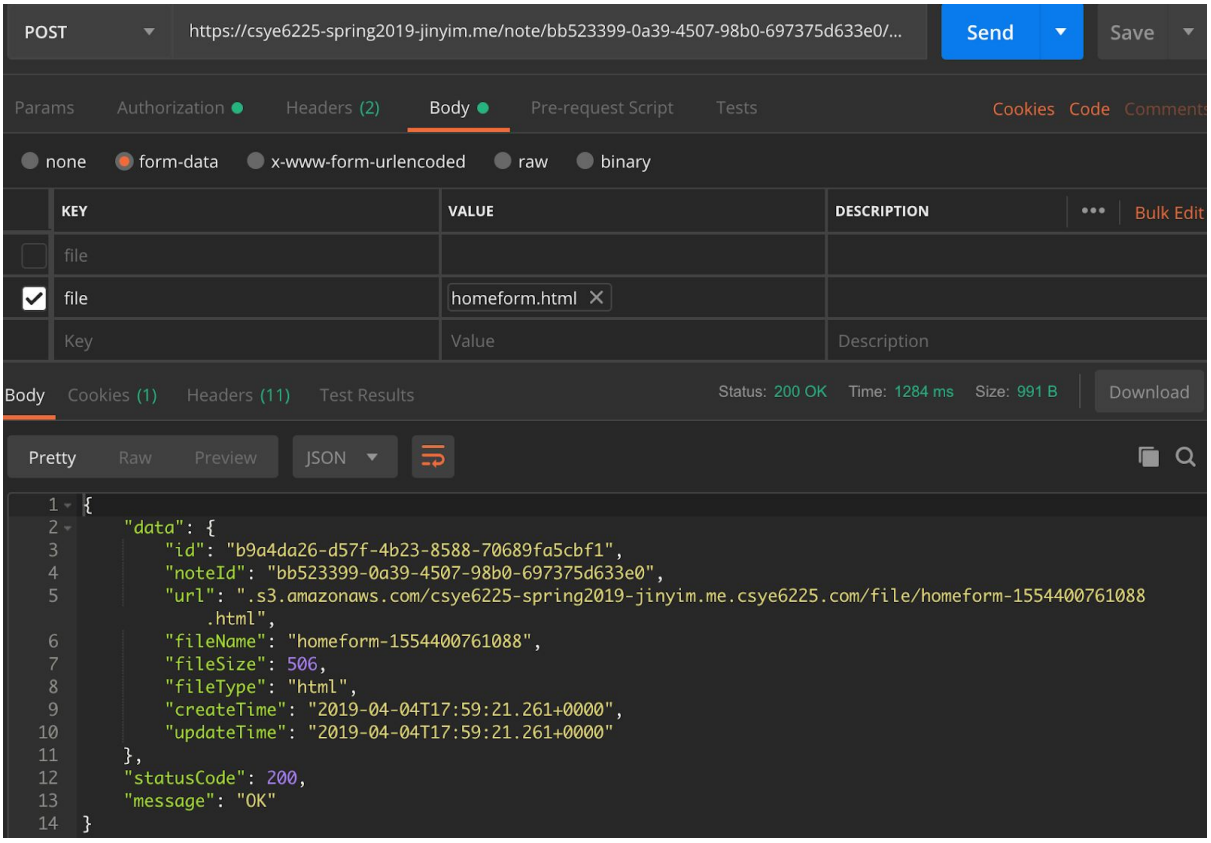
Attack Vector: Cross-Site Scripting (XSS) attacks

The reason why I choose this Attack Vector:

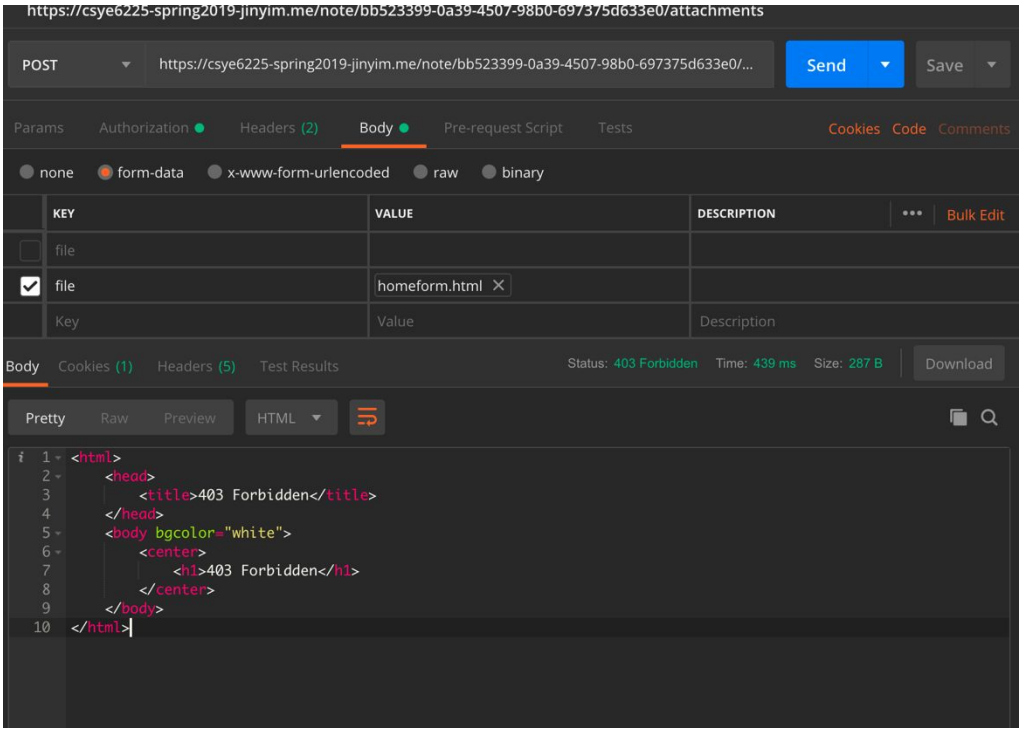
Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Attack Result:

Before I active aws waf:
I use postman to put a html file to my s3 bucket with my POST method in this url:https://csye6225-spring2019-jinyim.me/note/bb523399-0a39-4507-98b0-697375d633e0/attachments. I uploaded a html file named “homeform.html” and it uploaded successfully.



After I active aws waf:
I use postman to put the same html file to my s3 bucket with my POST method in the same url : https://csye6225-spring2019-jinyim.me/note/bb523399-0a39-4507-98b0-697375d633e0/attachments. This time the aws waf gives me a 403 forbidden which means aws waf works well.



3. Abnormal requests via size restrictions

Attack Vector：Abnormal requests via size restrictions

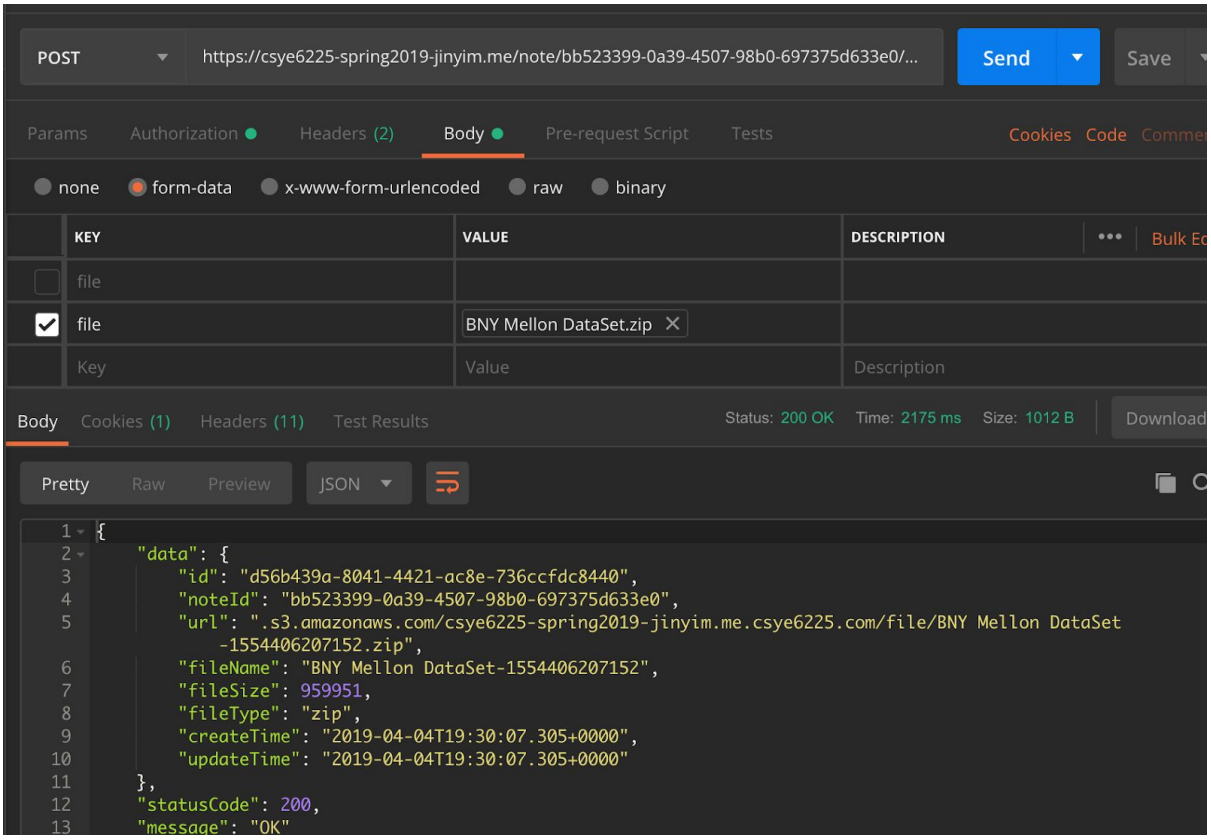
The reason why I choose this Attack Vector:

This is a very easy attack method and very usual from all kinds of request. Like Http flood which is a DDos attack. The attack is most effective when it forces the server or application to allocate the maximum resources possible in response to each single request. Thus, the perpetrator will generally aim to inundate the server or application with multiple requests that are each as processing-intensive as possible.For this reason HTTP flood attacks using POST requests tend to be the most resource-effective from the attacker's perspective; as POST requests may include parameters that trigger complex server-side processing. On the other hand, HTTP GET-based attacks are simpler to create, and can more effectively scale in a botnet scenario.

Attack Result:

Before I active aws waf:

I upload a big size file to my s3 bucket and it uploaded successfully. The file size is around 950,000 bytes.



After I active aws waf:

I upload the same big size file to my s3 bucket and it give me a 403 forbidden error which means waf works well!