Sri Lanka Institute of Information Technology

Specialized in Cyber Security

Year 2, Semester 2

IE2062 – Web Security

Bug Bounty – Report 04

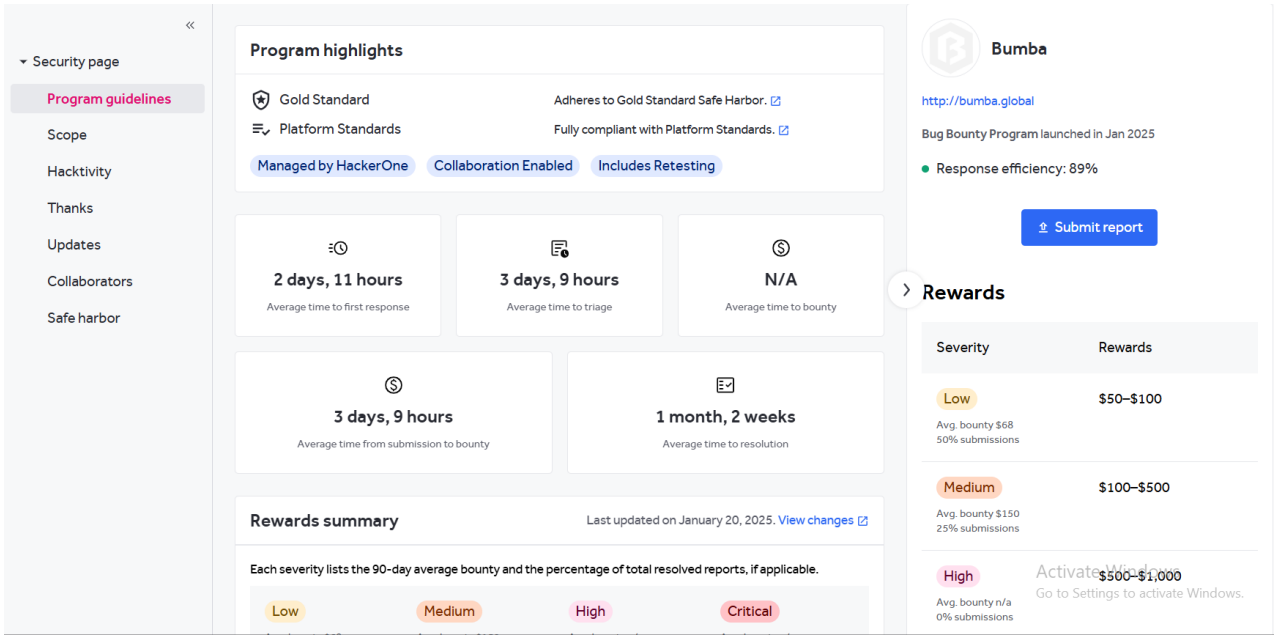
Student ID No.	Name
IT23136106	D.M.M. Pasindu Supushmika

Table of Contents

1. Website Overview

Bumba - Cryptocurrency trading platform

HackerOne Link: Bumba | Bug Bounty Program Policy | HackerOne

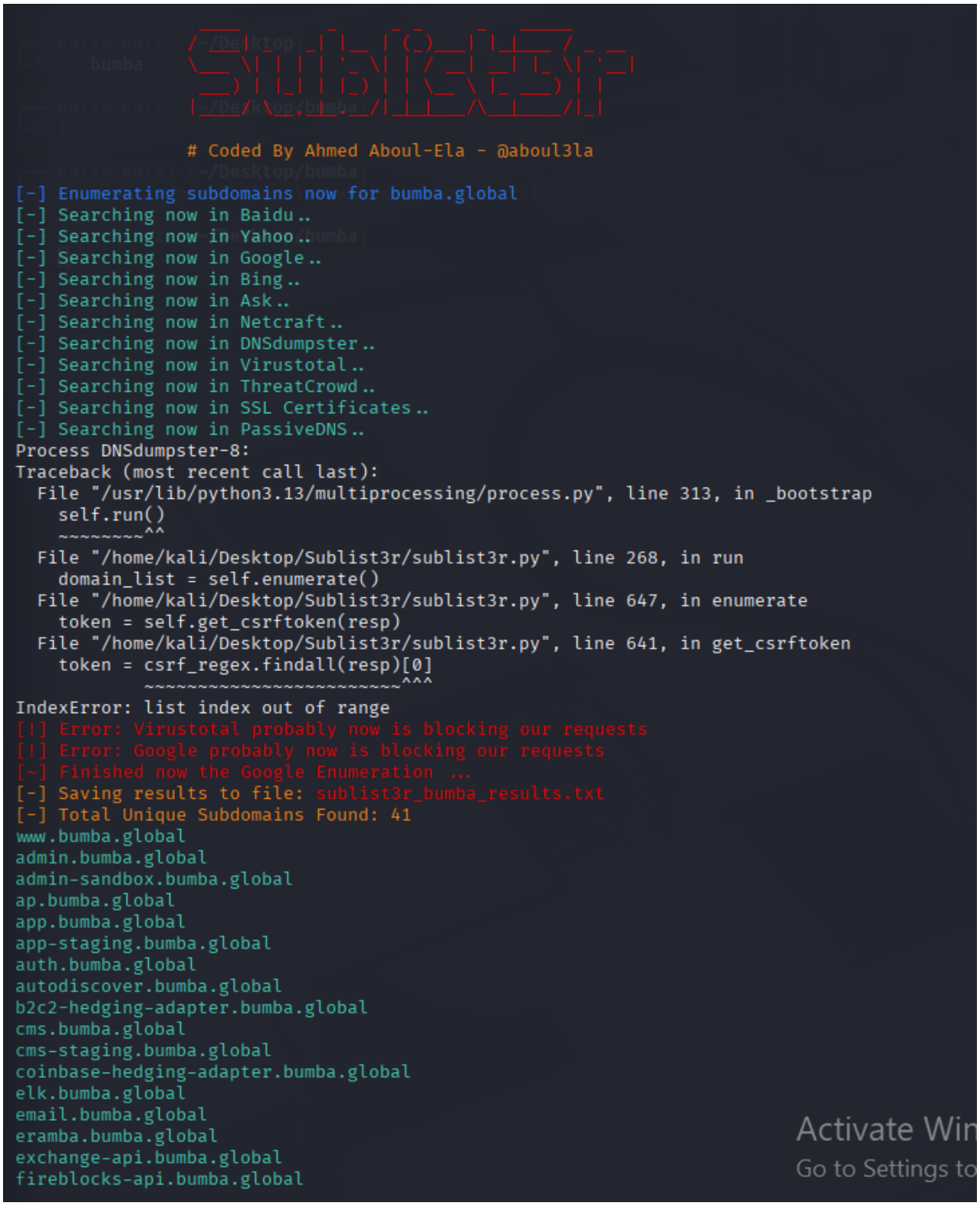


Step 01: Gather Information.

a. Sub-domain Discovery

i. Sublist3r: sublist3r_bumba_results.txt

Tool **:** Sublist3r
Code **:** python3 sublist3r.py -d bumba.globle -o sublist3r_bumba_results.txt
Explanation:
python3 sublist3r.py - Run the script using python
-d bumba.global - Target domain
-o sublist3r_bumba_results.txt – Output file where the result is saved



www.bumba.global
admin.bumba.global
admin-sandbox.bumba.global
ap.bumba.global
app.bumba.global
app-staging.bumba.global
auth.bumba.global
autodiscover.bumba.global
b2c2-hedging-adapter.bumba.global
cms.bumba.global
cms-staging.bumba.global
coinbase-hedging-adapter.bumba.global
elk.bumba.global
email.bumba.global
eramba.bumba.global
exchange-api.bumba.global
fireblocks-api.bumba.global
fireblocks-mainnet.bumba.global
fireblocks-testnet.bumba.global
fns-login.bumba.global
lyncdiscover.bumba.global
middleware-api.bumba.global
middleware-api-sandbox.bumba.global
middleware-api-sandbox1.bumba.global
mm.bumba.global
msoid.bumba.global
redmine.bumba.global
sandbox.bumba.global
sandbox-auth.bumba.global
sandbox1.bumba.global
sip.bumba.global
status.bumba.global
admin.status.bumba.global
tcms.bumba.global
tm.bumba.global
treasury.bumba.global
treasury-api.bumba.global
treasury-api-sandbox.bumba.global
treasury-api-v2.bumba.global
treasury-sandbox.bumba.global
treasury-sandbox-login.bumba.global

ii. Subfinder: subfinder_result_bumba.txt

Tool : Subfinder
Code : subfinder -d bumba.global -o subfinder_result_bumba.txt
Explanation:
subfinder - run subfinder too
l -d bumba.global - Mention the target website
-o subfinder_result_bumba.txt – Mention the output file

cms-staging.bumba.global
fireblocks-testnet.bumba.global
coinbase-hedging-adapter.bumba.global
eramba.bumba.global
cms.bumba.global
middleware-api.bumba.global
mm.bumba.global
bumba.global
sandbox-auth.bumba.global
fns-login.bumba.global
elk.bumba.global
sandbox1.bumba.global
redmine.bumba.global
sip.bumba.global
app.bumba.global
admin.status.bumba.global
auth.bumba.global
admin.bumba.global
tcms.bumba.global
sandbox.bumba.global
autodiscover.bumba.global
b2c2-hedging-adapter.bumba.global
tm.bumba.global
status.bumba.global
treasury-sandbox.bumba.global
treasury-api.bumba.global
fireblocks-mainnet.bumba.global
fireblocks-api.bumba.global
www.bumba.global
msoid.bumba.global
middleware-api-sandbox.bumba.global
treasury-api-sandbox.bumba.global
middleware-api-sandbox1.bumba.global
app-staging.bumba.global
treasury-api-v2.bumba.global
exchange-api.bumba.global
treasury.bumba.global
treasury-sandbox-login.bumba.global
admin-sandbox.bumba.global
ap.bumba.global
lyncdiscover.bumba.global
email.bumba.global

b. Live Subdomain Discovery

Tool : httpx: livesub_results.txt

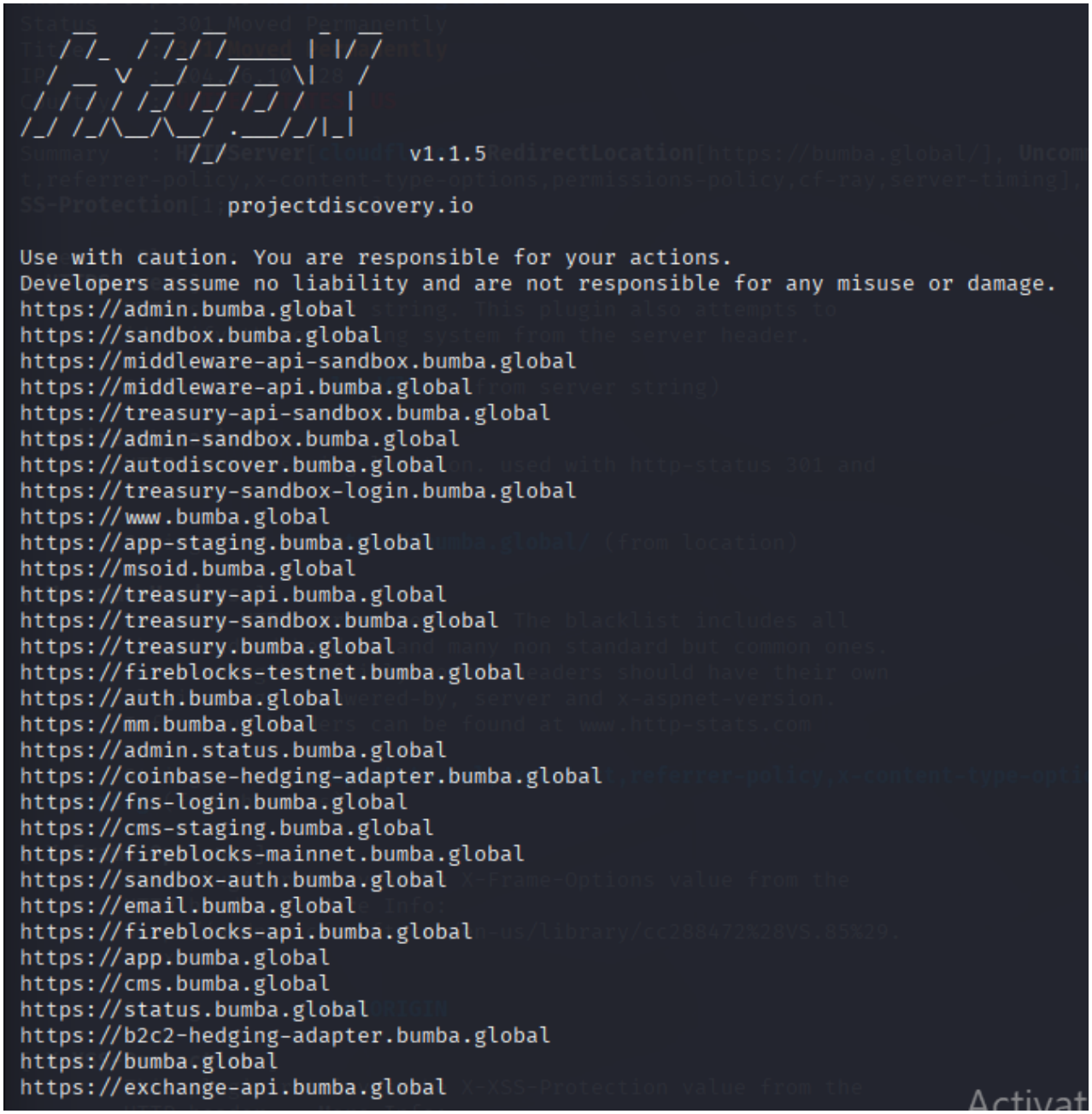
Code : httpx-toolkit -l subfinder_result_bumba.txt -o livesub_results.txt

Explanation:

httpx-toolkit - run the httpx tool

-l subfinder_result_bumba.txt – mention the file containing input

-o livesub_results.txt – mention the file which should write the output



https://admin.bumba.global
https://sandbox.bumba.global
https://middleware-api-sandbox.bumba.global
https://middleware-api.bumba.global
https://treasury-api-sandbox.bumba.global
https://admin-sandbox.bumba.global
https://autodiscover.bumba.global
https://treasury-sandbox-login.bumba.global
https://www.bumba.global
https://app-staging.bumba.global
https://msoid.bumba.global
https://treasury-api.bumba.global
https://treasury-sandbox.bumba.global
https://treasury.bumba.global
https://fireblocks-testnet.bumba.global
https://auth.bumba.global
https://mm.bumba.global
https://admin.status.bumba.global
https://coinbase-hedging-adapter.bumba.global
https://fns-login.bumba.global
https://cms-staging.bumba.global
https://fireblocks-mainnet.bumba.global
https://sandbox-auth.bumba.global
https://email.bumba.global
https://fireblocks-api.bumba.global
https://app.bumba.global
https://cms.bumba.global
https://status.bumba.global
https://b2c2-hedging-adapter.bumba.global
https://bumba.global
https://exchange-api.bumba.global

c. IP Discovery

Tool: nslookup: nslookup_result.txt

Code: since we whole file with subdomains, to find IP addresses using "nslookup" we need to make a loop until all the Ips of all the subdomains are found.

```
while read sub; do
 echo "Looking up: $sub" >> nslookup_result.txt
  nslookup "$sub" | awk '/^Name:|^Address:/' >> nslookup_result.txt
  echo "------------------------" >> nslookup_result.txt
done < livesub_results.txt
```

Explanation:

While read sub; do - start of the loop

Echo "Looking up: $sub">>nslookup_result.txt - print message "Looking up: subdomain" into the file "*nslookup_result*.txt"

nslookup "$sub" | awk '/^Name:|^Address:/' >> nslookup_result.txt - run the nslookup command

echo "_____" >> nslookup_result.txt - separate one subdomain details from another

done < livesub_results.txt - End the loop and continue until the lines in the livesub_results.txt

```
┌──(kali㉿kali)-[~/Desktop/bumba]
└─$ ./nslookup_script.sh

┌──(kali㉿kali)-[~/Desktop/bumba]
└─$ cat nslookup_result.txt
Looking up: https://admin.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://sandbox.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://middleware-api-sandbox.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://middleware-api.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://treasury-api-sandbox.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://admin-sandbox.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://autodiscover.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://treasury-sandbox-login.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://www.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://app-staging.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://msoid.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://treasury-api.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://treasury-sandbox.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://treasury.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://fireblocks-testnet.bumba.global
Address:        192.168.0.1#53
------------------------
Looking up: https://auth.bumba.global
Address:        192.168.0.1#53
------------------------
```

IP list:

Looking up: https://admin.bumba.global
Address: 192.168.0.1#53

Looking up: https://sandbox.bumba.global
Address: 192.168.0.1#53

Looking up: https://middleware-api-sandbox.bumba.global
Address: 192.168.0.1#53

Looking up: https://middleware-api.bumba.global
Address: 192.168.0.1#53

Looking up: https://treasury-api-sandbox.bumba.global
Address: 192.168.0.1#53

Looking up: https://admin-sandbox.bumba.global
Address: 192.168.0.1#53

Looking up: https://autodiscover.bumba.global
Address: 192.168.0.1#53

Looking up: https://treasury-sandbox-login.bumba.global
Address: 192.168.0.1#53

Looking up: https://www.bumba.global
Address: 192.168.0.1#53

Looking up: https://app-staging.bumba.global
Address: 192.168.0.1#53

Looking up: https://msoid.bumba.global
Address: 192.168.0.1#53

Looking up: https://treasury-api.bumba.global
Address: 192.168.0.1#53

Looking up: https://treasury-sandbox.bumba.global
Address: 192.168.0.1#53

Looking up: https://treasury.bumba.global
Address: 192.168.0.1#53

Looking up: https://fireblocks-testnet.bumba.global
Address: 192.168.0.1#53

Looking up: https://auth.bumba.global
Address: 192.168.0.1#53

Looking up: https://mm.bumba.global
Address: 192.168.0.1#53
---

d. Open Ports

Tool: nmap: nmap_result.txt

Code: nmap -sV -A -v -O bumba.global -oN nmap_results.txt

Explanation:

nmap - start the tool
-sV - Service and version detection
-A - OS detection, version detection, script scanning
-v - increase verbosity level
-O - Os detection
- *bumba.global* - target website
-oN nmap_results.txt - result in an output text file

```
┌──(kali㉿kali)-[~/Desktop/bumba]
└─$ nmap -sV -A -v -O bumba.global -oN nmap_result.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-27 19:05 +0530
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:05
Completed NSE at 19:05, 0.00s elapsed
Initiating NSE at 19:05
Completed NSE at 19:05, 0.00s elapsed
Initiating NSE at 19:05
Completed NSE at 19:05, 0.00s elapsed
Initiating Ping Scan at 19:05
Scanning bumba.global (172.67.69.169) [4 ports]
Completed Ping Scan at 19:05, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:05
Completed Parallel DNS resolution of 1 host. at 19:05, 0.18s elapsed
Initiating SYN Stealth Scan at 19:05
Scanning bumba.global (172.67.69.169) [1000 ports]
Discovered open port 25/tcp on 172.67.69.169
Discovered open port 443/tcp on 172.67.69.169
Discovered open port 8080/tcp on 172.67.69.169
Discovered open port 80/tcp on 172.67.69.169
Completed SYN Stealth Scan at 19:05, 8.16s elapsed (1000 total ports)
Initiating Service scan at 19:06
Scanning 4 services on bumba.global (172.67.69.169)
Completed Service scan at 19:06, 15.96s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against bumba.global (172.67.69.169)
Retrying OS detection (try #2) against bumba.global (172.67.69.169)
Initiating Traceroute at 19:06
Completed Traceroute at 19:06, 0.08s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 19:06
Completed Parallel DNS resolution of 2 hosts. at 19:06, 0.13s elapsed
NSE: Script scanning 172.67.69.169.
Initiating NSE at 19:06
Completed NSE at 19:06, 32.12s elapsed
Initiating NSE at 19:06
Completed NSE at 19:07, 31.73s elapsed
Initiating NSE at 19:07
Completed NSE at 19:07, 0.00s elapsed
Nmap scan report for bumba.global (172.67.69.169)
Host is up (0.014s latency).
Other addresses for bumba.global (not scanned): 104.26.10.128 104.26.11.128 2606:4700:20::681a:a80 2606:4700:20::6
1a:b80 2606:4700:20::ac43:45a9
Not shown: 996 filtered tcp ports (no-response)
PORT     STATE SERVICE    VERSION
25/tcp   open  tcpwrapped
|_smtp-commands: SMTP EHLO bumba.global: failed to receive data: connection closed
80/tcp   open  tcpwrapped
| http-methods:
|_  Supported Methods: OPTIONS
|_http-title: Did not follow redirect to https://bumba.global/
443/tcp  open  https?
|_http-title: 400 The plain HTTP request was sent to HTTPS port
| http-methods:
|_  Supported Methods: POST
```

e. Used Technologies

Tool: whatweb - [whatweb_results.txt]

Code: whatweb -v bumba.global > whatweb_result.txt

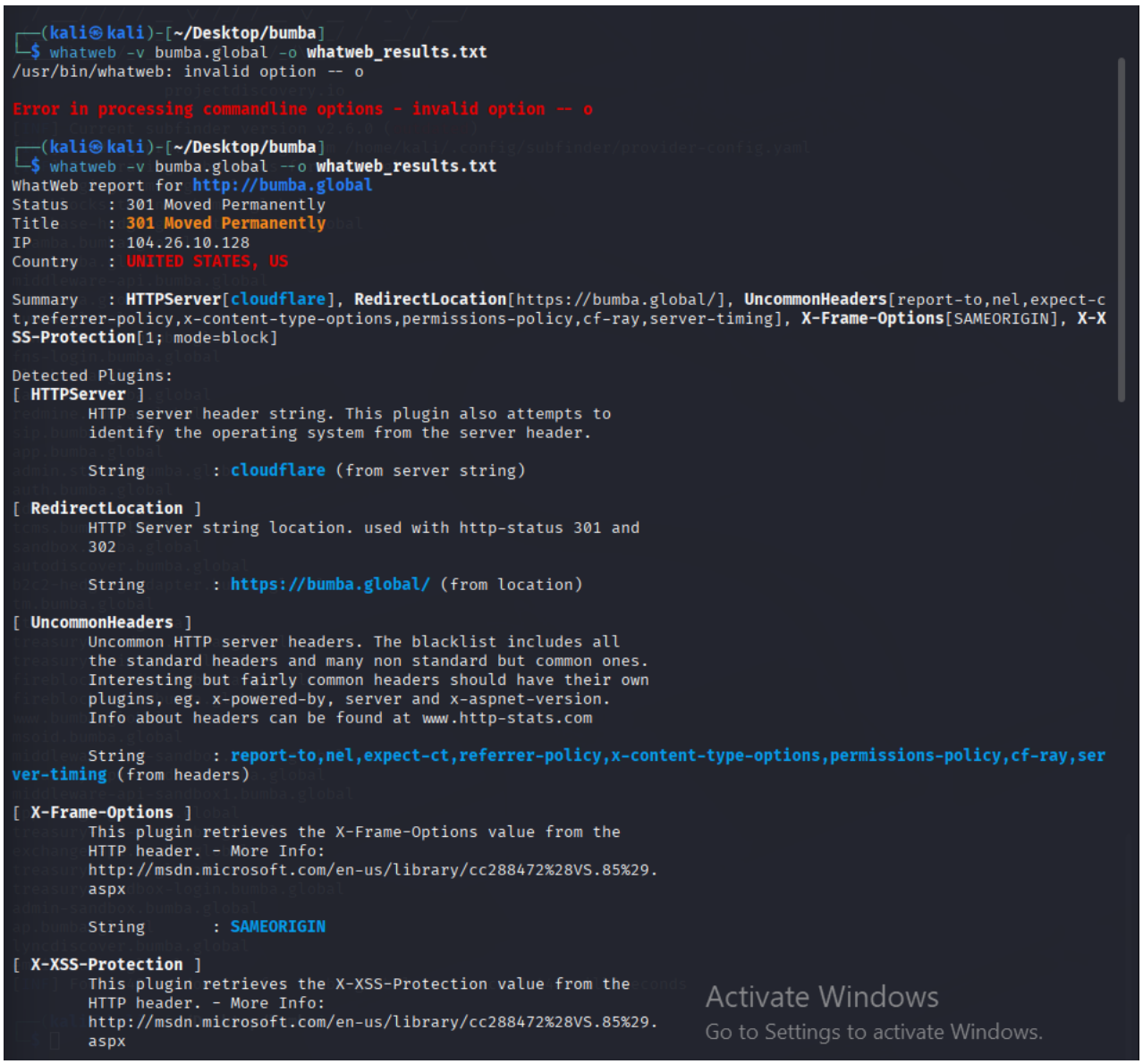
Explanation:

whatweb - start whatweb tool

-v - verbose

Bumba.global - target website

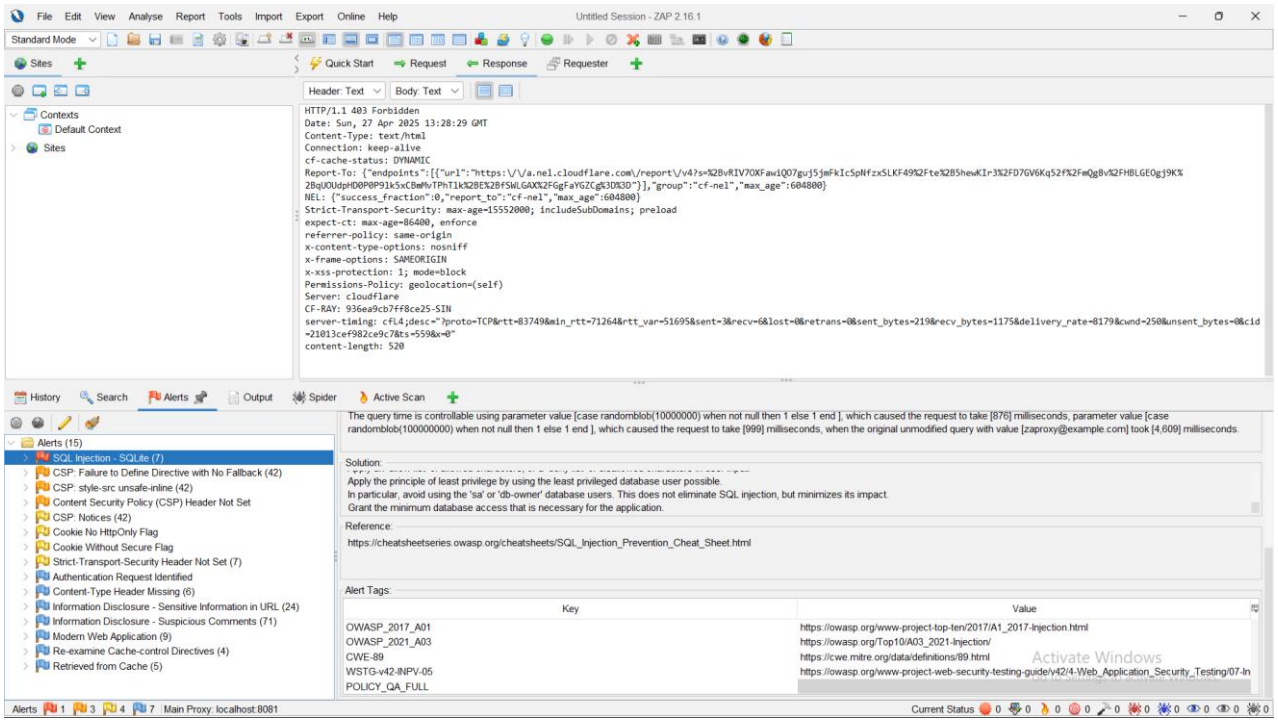
> whatweb_result.txt - file with the output



3. Step 02: Scanning and vulnerability identification

a. Identify Potential Vulnerabilities

Tool : OWASP ZAP
Vulnerability : SQL Injection - SQLite



SQL Injection - SQLite:

URL: https://bumba.global/en/market?email=zaproxy%40example.com
Risk: High
Confidential: Medium
Parameter: email
Attack: case randomblob(10000000) when not null then 1 else 1 end
Evidence: The query time is controllable using parameter value [case randomblob(10000000) when not null then 1 else 1 end], which caused the request to take [876] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [999] milliseconds, when the original unmodified query with value [zaproxy@example.com] took [4,609] milliseconds.
CWE ID: 89
WASC ID: 19
Source: Active (40024 - SQL Injection - SQLite)
Input Vector: URL Query String

- Description: SQL injection may be possible..
- Other Info: The query time is controllable using parameter value [case randomblob(10000000) when not null then 1 else 1 end], which caused the request to take [876] milliseconds, parameter value [case randomblob(100000000) when not null then 1 else 1 end], which caused the request to take [999] milliseconds, when the original unmodified query with value [zaproxy@example.com] took [4,609] milliseconds.
- 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 Prepared Statement or Callable Statement, 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 an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input. Apply the principle 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..

- Reference:
  - https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- Alert Tags:
  - OWASP_2017_A01: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html
  - OWASP_2021_A03: https://owasp.org/Top10/A03_2021-Injection/
  - CWE-89: https://cwe.mitre.org/data/definitions/89.html
  - WSTG-v42-INPV-05: https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection
  - POLICY_QA_FULL

b. SQL Injection - SQLite

SQL Injection – SQLite refers to a type of vulnerability where an attacker manipulates SQL queries by injecting malicious input into a web application's database interactions, specifically targeting SQLite databases. Because SQLite is often used in lightweight applications and mobile apps, developers may underestimate security risks, making the system vulnerable to unauthorized data access, data modification, or even full database compromise.

.

Cause of SQL Injection – SQLite website:

- Directly embedding unsanitized user input into SQL queries
- Failure to use parameterized queries or prepared statements
- Lack of input validation or weak sanitization techniques
- Misconfigured database access permissions allowing over-privileged queries
- Improper error handling that reveals database structure or query syntax
- Relying on client-side validation without enforcing it on the server side

Propositions to Mitigation or Fix:

- Use Parameterized Queries: Always use prepared statements with placeholders for user inputs
- Validate and Sanitize Inputs: Strictly validate all incoming data based on expected format and type
- Employ Least Privilege Principle: Limit database user permissions to only what is necessary
- Error Handling: Do not display detailed database errors to users; log them securely instead
- Use ORM Libraries: Consider using trusted Object-Relational Mapping (ORM) frameworks that abstract query building
- Perform Regular Security Testing: Conduct code reviews, vulnerability scanning, and penetration testing focused on injection flaws
- Keep SQLite Updated: Use the latest stable version of SQLite, as older versions might have security vulnerabilities

4. Step 03: Exploitation and Validation

Request:

```
GET https://bumba.global/en/market?email=case+randomblob%2810000000%29+when+not+null+then+1+else+1+end+ HTTP/1.1
host: bumba.global
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: https://bumba.global/en/market
Cookie: Next-Locale=en
```

Response:

```
HTTP/1.1 403 Forbidden
Date: Sun, 27 Apr 2025 13:28:29 GMT
Content-Type: text/html
Connection: keep-alive
cf-cache-status: DYNAMIC
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=%2BvRIV7OXFawiQO7guj5jmFkIcSpNfzxSLKF49%2Fte%2B5hewKIr3%2FD7GV6Kq52f%2FmQg8v%2FHBLGEOgj9K%
2BqUOUdpHD0P0P91k5xCBmMvTPhT1k%2BE%2BfSWLGAX%2FGgFaYGZCg%3D%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
expect-ct: max-age=86400, enforce
referrer-policy: same-origin
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
Permissions-Policy: geolocation=(self)
Server: cloudflare
CF-RAY: 936ea9cb7ff8ce25-SIN
server-timing: cfL4;desc="?proto=TCP&rtt=83749&min_rtt=71264&rtt_var=51695&sent=3&recv=6&lost=0&retrans=0&sent_bytes=219&recv_bytes=1175&delivery_rate=8179&cwnd=250&unsent_bytes=0&cid
=21013cef982ce9c7&ts=559&x=0"
content-length: 520
```

5. Step 04: Mitigation / Fix

Immediate Mitigation Actions:

1. Block the vulnerable endpoints or disable until fixed.
2. Add SQL injection filters.

Secure Coding:

1. Use parameterized Queries
2. Input Validation – Only permit expected email formats and block suspicious Keywords.
3. Ensure DB users have least privilege.

Long Term Prevention:

1. Use Object-Relational Mapping Libraries (tools that allow developers to interact with a relational using object-oriented programming languages) to avoid raw SQL.
2. Developer Training