Sri Lanka Institute of Information Technology

BUG BOUNTY REPORT 07

(Ring.com Web site)

IE2062 – Web Security

W.A.K.S Wijethunga

IT23361768

Table of Contents

1. Introduction to bug bounty program and audit scope

❖ Airbnb.com

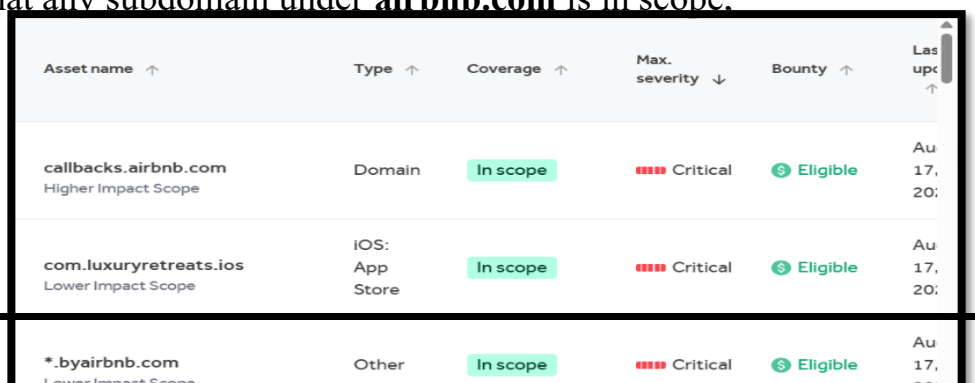
Airbnb.com is the official web platform of Airbnb, Inc., a global marketplace for lodging, homestays, and tourism experiences. The website allows users to list, discover, and book accommodations around the world. It also supports user interactions, transactions, profile management, and a wide range of service features for both hosts and guests.

This bug bounty report documents a security vulnerability discovered on Airbnb.com that may affect the confidentiality, integrity, or availability of the platform or its users. The vulnerability was identified through responsible research and submitted in accordance with Airbnb's disclosure guidelines.

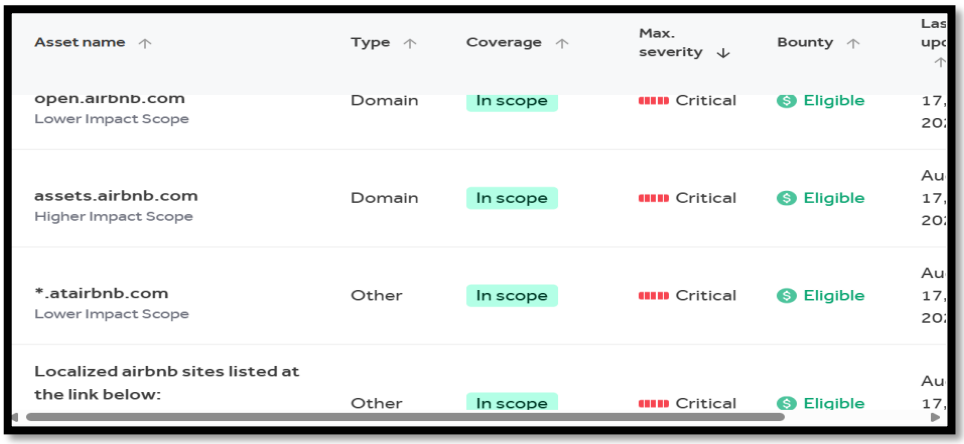
In Hackerone bug bounty program, they defined these subdomains (and all inclusive) as valid subdomains for testing.

- **callbacks.airbnb.com**
- **open.airbnb.comJs**
- **assets.airbnb.com**
- **next.airbnb.com**
- **one.airbnb.com**
- **www.hoteltonight.com**
- **api.airbnb.com**
- **admin.demo.urbandoor.com**
- **luckeyhomes.com**
- **luckey.fr**

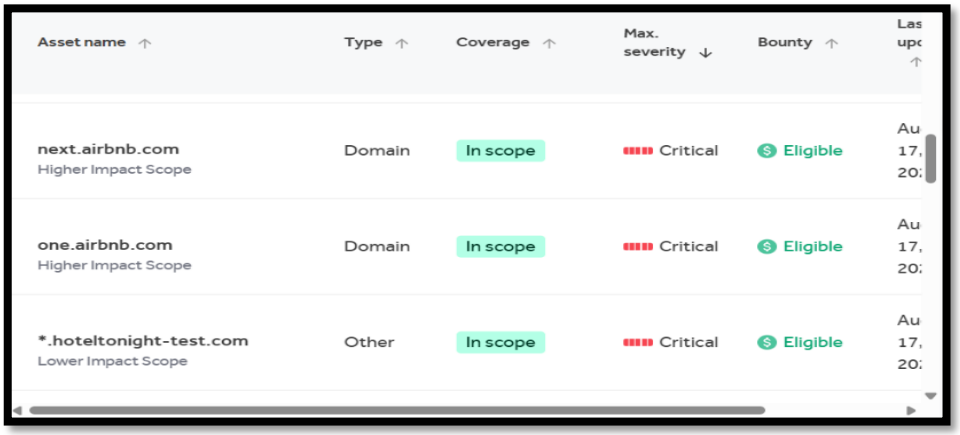
Eligible in-scope subdomains for bug bounty program are mentioned below and they mention that any subdomain under **airbnb.com** is in scope.



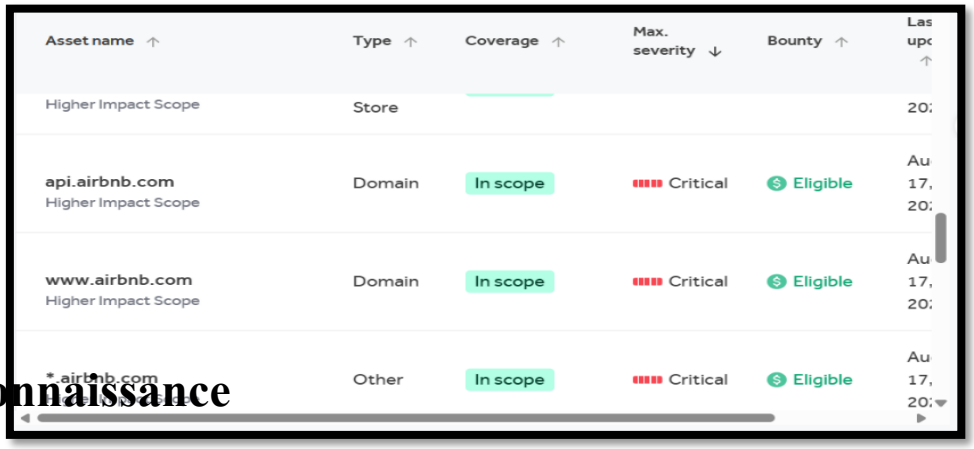
Asset name	Type	Coverage	Max. severity	Bounty	Las upd
callbacks.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
com.luxuryretreats.ios Lower Impact Scope	iOS: App Store	In scope	Critical	Eligible	Au 17, 20:
*.byairbnb.com	Other	In scope	Critical	Eligible	Au 17,



Asset name	Type	Coverage	Max. severity	Bounty	Las upd
open.airbnb.com Lower Impact Scope	Domain	In scope	Critical	Eligible	17, 20:
assets.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
*.atairbnb.com Lower Impact Scope	Other	In scope	Critical	Eligible	Au 17, 20:
Localized airbnb sites listed at the link below:	Other	In scope	Critical	Eligible	Au 17,



Asset name	Type	Coverage	Max. severity	Bounty	Las upd
next.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
one.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
*.hoteltonight-test.com Lower Impact Scope	Other	In scope	Critical	Eligible	Au 17, 20:



Asset name	Type	Coverage	Max. severity	Bounty	Las upd
Higher Impact Scope	Store				20:
api.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
www.airbnb.com Higher Impact Scope	Domain	In scope	Critical	Eligible	Au 17, 20:
*.airbnb.com	Other	In scope	Critical	Eligible	Au 17, 20:

2. Reconnaissance

The goal of this reconnaissance is to gather information about the **web.crypto.com** website, including its infrastructure, technologies, and potential security posture. This information will help identify potential vulnerabilities and attack vectors.

I. Find Domain using Sublist3r Tool

Sublist3r, a Python-based tool, is designed to discover subdomains associated with a specified target website. Leveraging search engines and online web services, it scours the web for available subdomains linked to the designated target domain. Given the freedom to scrutinize any subdomain under reddit.com, it's prudent to identify additional subdomains for testing purposes.

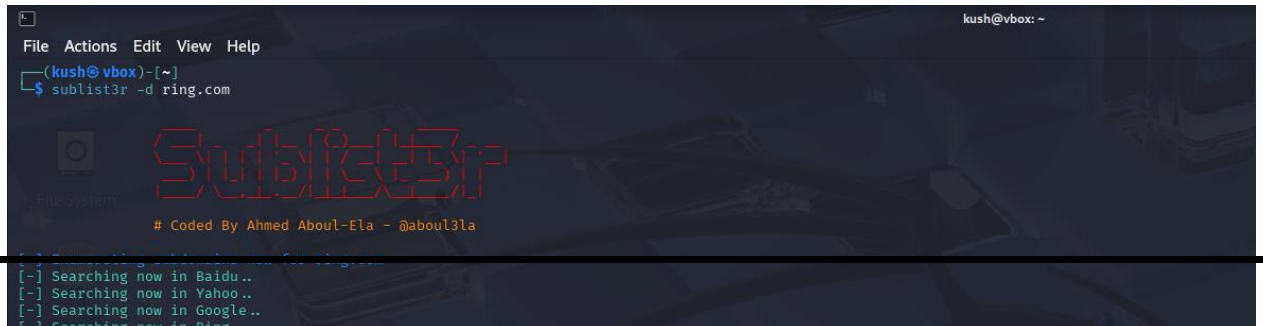
To install Sublist3r, navigate to its GitHub repository at https://github.com/aboul3la/Sublist3r.git. This repository hosts all the necessary files required for installing the tool. Execute the following command in your shell to download it:

```
***git clone https://github.com/aboul3la/Sublist3r.git***
```

Please note that Sublist3r necessitates either Python 2.7 or Python 3.4 to operate smoothly.
After downloading the files, go inside the 'Sublist3r' directory and install the requirements by entering,

sudo pip install -r requirements.txt

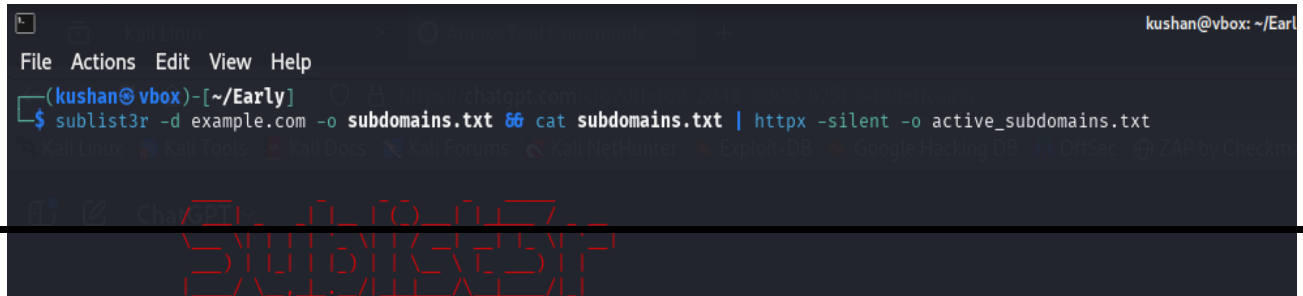
After installing the requirements, enter
sublist3r -d airbnb.com -o subdomains.txt
to find subdomains under the mentioned domain.



Upon examining for accessible subdomains, the next step involves identifying those that are operational. This can be accomplished by employing an additional tool known as '**httpx**'.

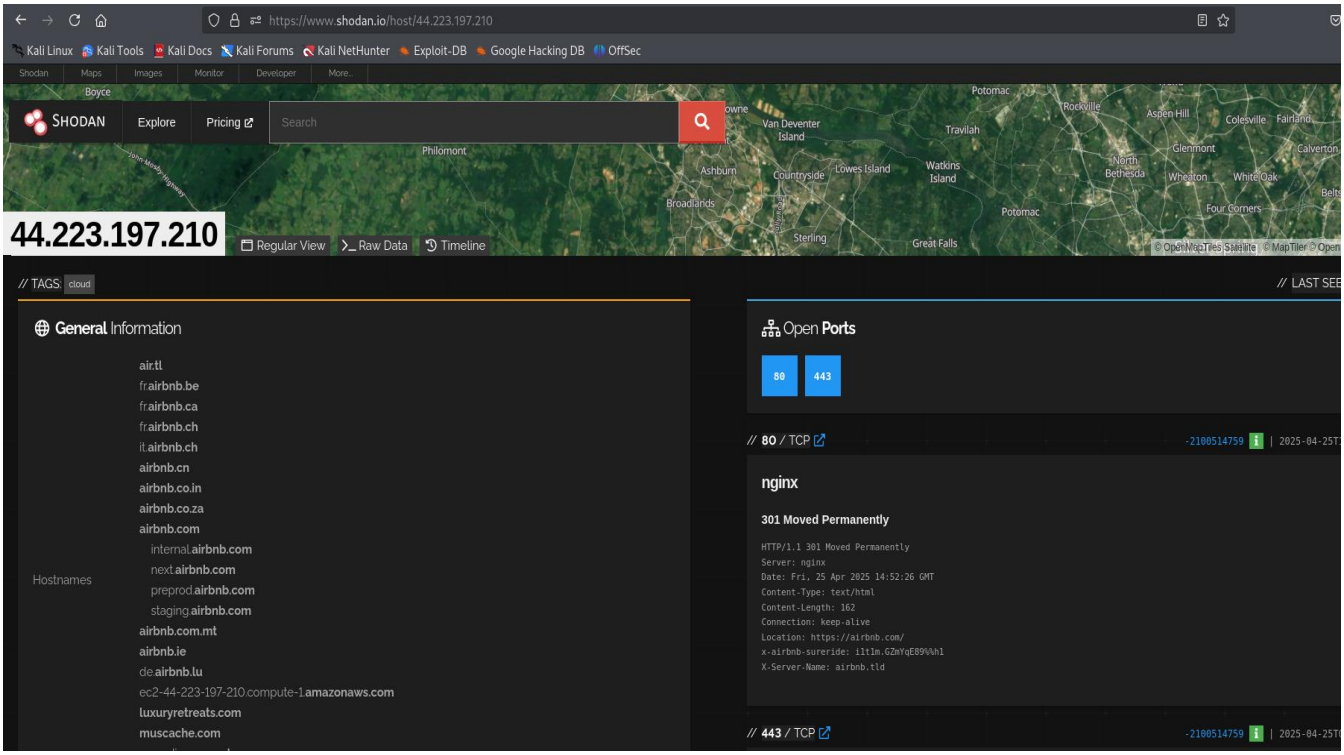
This tool can find domains that are up and running. To find active subdomains under this site, I am using the text file generated before by the sublist3r and writing the active subdomains to another new file.

Following the completion of the scan, the findings reveal that the majority of the subdomains are indeed active.



II. Identify exposed services using Shodan

Shodan is a potent search engine made to look through and index gadgets that are linked to the internet. Shodan concentrates on hardware, such as servers, routers, and Internet of Things devices, as well as services, such as web servers, databases, and remote access tools, in contrast to standard search engines that crawl websites. It is a useful tool for security researchers, penetration testers, and bug bounty hunters since it gathers metadata from these devices, such as banners, open ports, and software versions. Shodan can be used to find exposed services that could be at danger to the organization due to misconfigured or attack-prone settings.



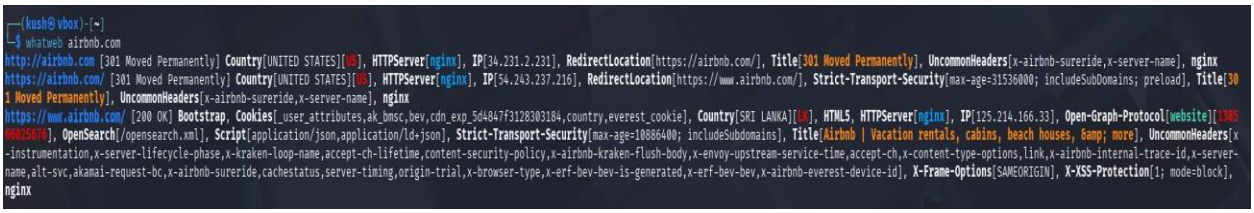
III. Detect technologies using Whatweb

Whatweb is a powerful open-source tool designed to identify the technologies used by websites. It works by analyzing the responses from a web server, such as HTTP headers, HTML content, cookies, and scripts, to detect the underlying technologies.

To detect technologies used by a website, simply run :

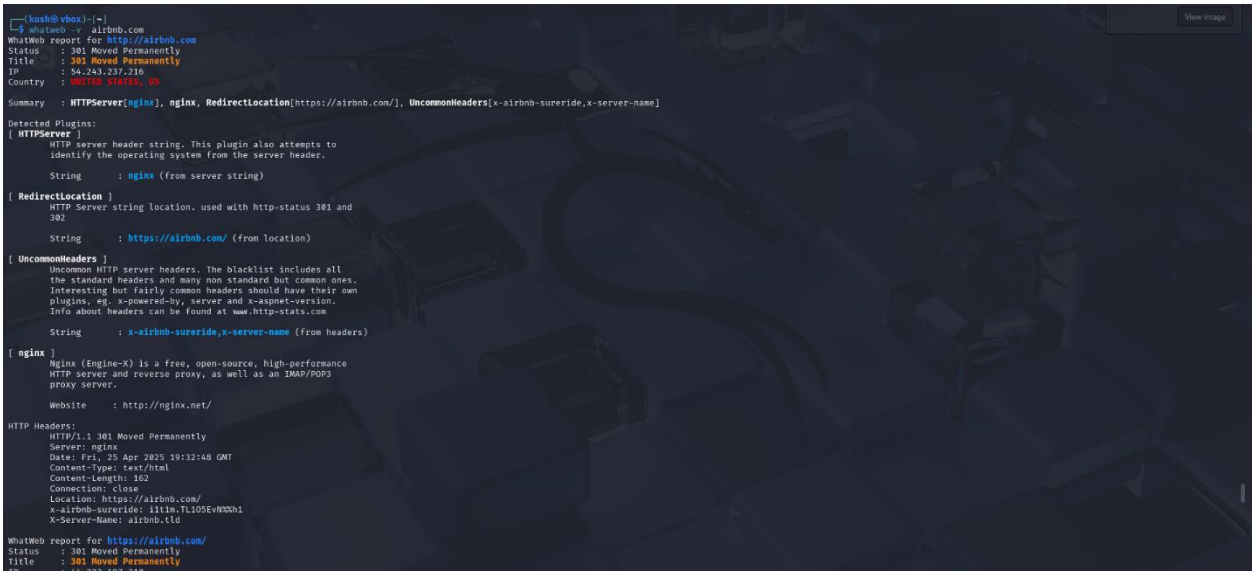
whatweb airbnb.com

This command will analyze the website and display a summary of the detected technologies**.**



To get detailed information about the detection process:

whatweb -v airbnb.com



3. Scanning Vulnerability Identifies

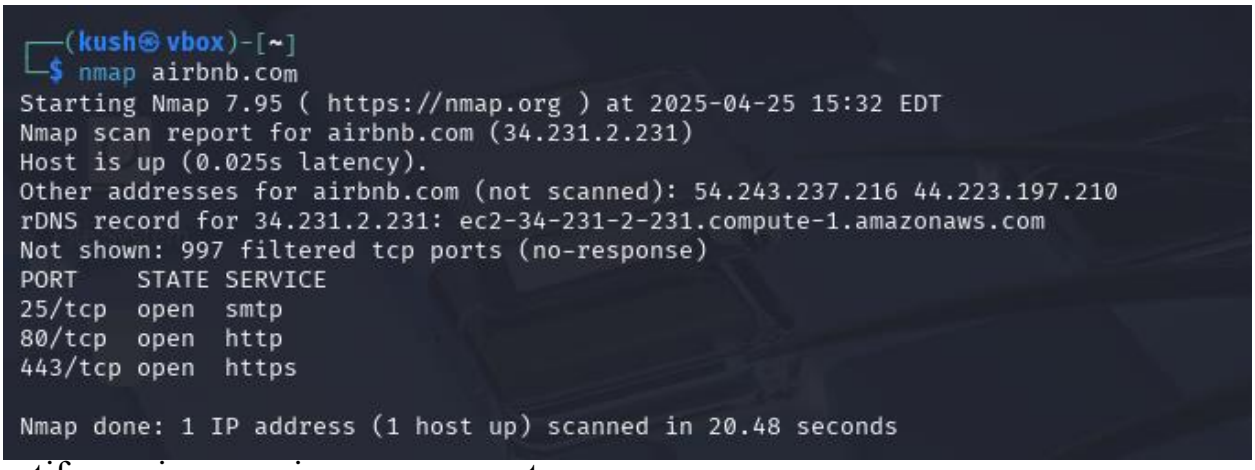
One of the most important steps in finding security flaws in a system, network, or application is vulnerability scanning. It entails identifying known vulnerabilities,

configuration errors, and possible attack routes using automated technologies. The objective is to evaluate the target's security posture and offer practical advice to reduce risks. For this, tools **like Nessus, OpenVAS, Nikto**, and **Nmap** are frequently utilized. In order to find vulnerabilities like out-of-date software, shoddy setups, or exposed sensitive data, the procedure involves scanning open ports, services, and applications.

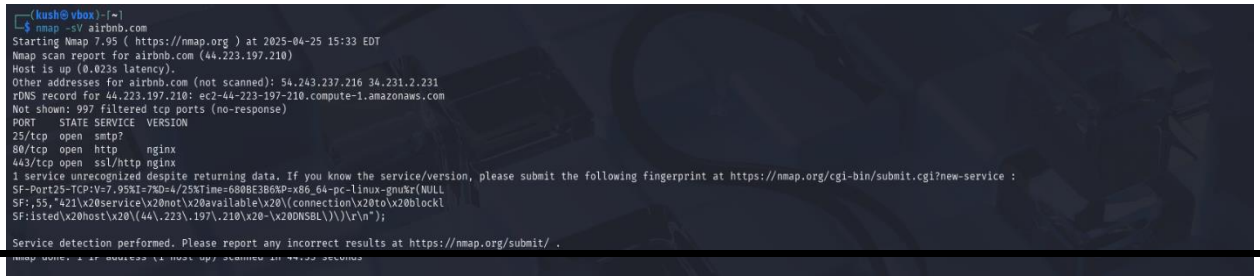
i. Open ports services

Nmap (Network Mapper) is a powerful tool for scanning open ports and identifying running services on a target system. By using the **nmap -sV** command, you can detect the version of services running on open ports, helping assess potential vulnerabilities. The -p- option scans all 65,535 ports, while -A enables OS detection, version detection, script scanning, and traceroute for a comprehensive analysis. The results typically display open ports, their associated services, and potential security risks, making it an essential tool for penetration testers and system administrators.

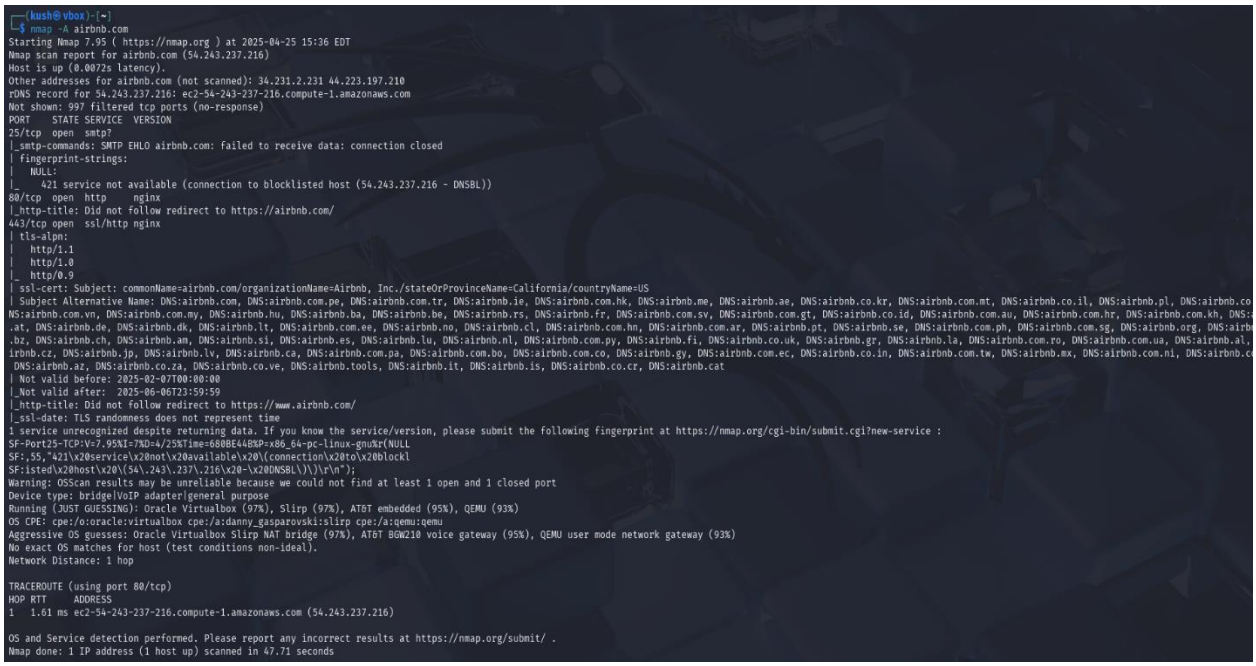
Scan the most commonly used on **web.crypto.com,**



Identify services running on open ports,



To get more detailed information, including **operating system detection**



ii. Web vulnerabilities

Nikto is an open-source web server scanner designed to identify vulnerabilities, outdated software, and security misconfigurations on web servers. It performs comprehensive testing for over 6700

vulnerabilities, including misconfigured files, outdated server software, and security holes.

Nikto -h airbnb.com using this command will scan zellepay.force.com for vulnerabilities, misconfigurations, and security issues.

Scans both HTTP and HTTPS,

nikto -h https://airbnb.com -ssl using this command runs a **Nikto** scan on https://zellepay.force.com while explicitly forcing SSL/TLS encryption.

```
┌──(kush㉿vbox)-[~]
└─$ nikto -h airbnb.com -ssl
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 44.223.197.210, 34.231.2.231, 54.243.237.216
+ Target IP:          44.223.197.210
+ Target Hostname:    airbnb.com
+ Target Port:        443
---------------------------------------------------------------------------
+ SSL Info:        Subject:  /C=US/ST=California/L=San Francisco/O=Airbnb, Inc./CN=airbnb.com
                   Ciphers:  ECDHE-RSA-AES256-GCM-SHA384
                   Issuer:   /C=US/O=DigiCert Inc/CN=DigiCert Global G2 TLS RSA SHA256 2020 CA1
+ Start Time:         2025-05-05 01:30:33 (GMT-4)
---------------------------------------------------------------------------
+ Server: nginx
+ /: Retrieved x-server-name header: airbnb.tld.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: Uncommon header 'x-airbnb-sureride' found, with contents: i1t1m.F1T8PGCl%%h1.
+ /: Uncommon header 'x-server-name' found, with contents: airbnb.tld.
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: https://www.airbnb.com/
```

Automated Testing

For automated testing, I've selected OWASP ZAP widely used tool within the industry.

OWASP ZAP

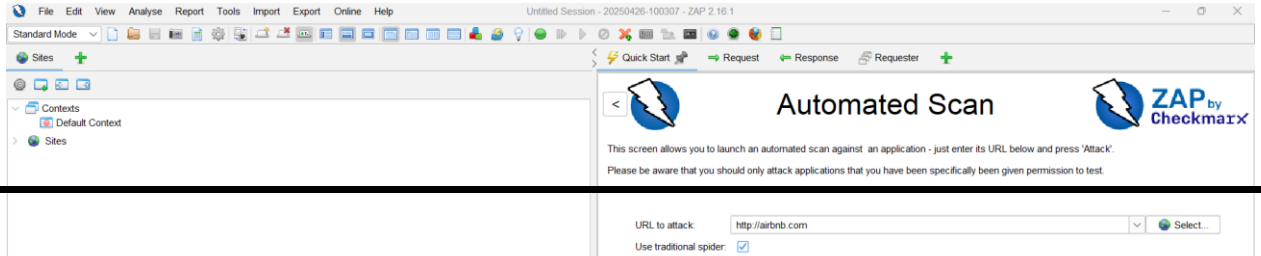
The Open Web Application Security Project Zed Attack Proxy (OWASP ZAP) is an open-source vulnerability scanner renowned for its capability to function as a Man-in-the-Middle (MITM) proxy. It assesses various vulnerabilities by scrutinizing responses from the web application or server. Notably convenient to utilize, OWASP ZAP offers customization options through the installation of modules, enabling efficient management of results.

Within this proxy, there are primarily two scan types available:

1. Automated Scan: Users input the target URL and initiate the attack. The behavior can be tailored by selecting the ZAP mode. This triggers all scripts against the target to detect vulnerabilities and generates reports accordingly.

2. Manual Explore: Users can navigate to the target web application and commence exploration. During manual exploration, ZAP HUD (Heads Up Display) captures each page, while the ZAP proxy records responses.

For this assessment, I am running ZAP on automated mode.



After specifying the target URL in the designated textbox, simply select "Attack" to initiate the scanning process. Upon completion, a comprehensive report of the findings can be generated by selecting "Report." Below are screenshots showcasing the results obtained after scanning several domains.

iii. Web server misconfigurations

Detailed Analysis of Missing Security Headers

1. **Missing X-Frame-Options Header**

Risk: The absence of the X-Frame-Options header makes the website potentially vulnerable to **clickjacking attacks**.

Impact:

- An attacker can embed the website inside an invisible or disguised <**iframe**> on a malicious page.
- Users may unknowingly interact with hidden UI elements (ex:-clicking buttons that perform unintended actions like fund transfers or password changes).
- This could lead to unauthorized transactions, account takeovers, or phishing scams if sensitive actions are exposed.

2. **Missing X-Content-Type-Options Header**

Risk: Without this header, browsers may perform **MIME sniffing**, which can lead to:

- **Cross-Site Scripting (XSS)**: If a file (ex:- an uploaded image) is misinterpreted as executable code.
- **Content Spoofing**: Attackers could disguise malicious scripts as harmless files (ex:- .jpg executing as JavaScript).

Impact:

- Exploitable in file upload features or improperly served static content.
- Could allow attackers to bypass security filters and execute malicious scripts in the context of the website.

Detailed Analysis of Cookie Security Issues

1. **IP Disclosure in Cookies**

Risk: When internal or non-routable IP addresses are included in cookies, it may lead to:

- Information Disclosure: Revealing internal infrastructure details such as private IPs, proxy chains, or internal network topology.

- Reconnaissance Advantage: Attackers can use disclosed IPs to map backend systems and better plan attacks (e.g., targeting specific ranges).

- SSRF Aid: In Server-Side Request Forgery scenarios, leaked internal IPs help attackers craft precise payloads to target internal services.

Impact:

- Exploitable in Cookie-Based Headers: If cookies such as __cf_bm, _cfuvid, or custom debug cookies store internal IP addresses, attackers can extract them via passive observation or via XSS.

- Network Enumeration: Leaked IPs might expose load balancer, proxy, or origin server addresses, assisting attackers in bypassing cloud protections or conducting targeted internal attacks.

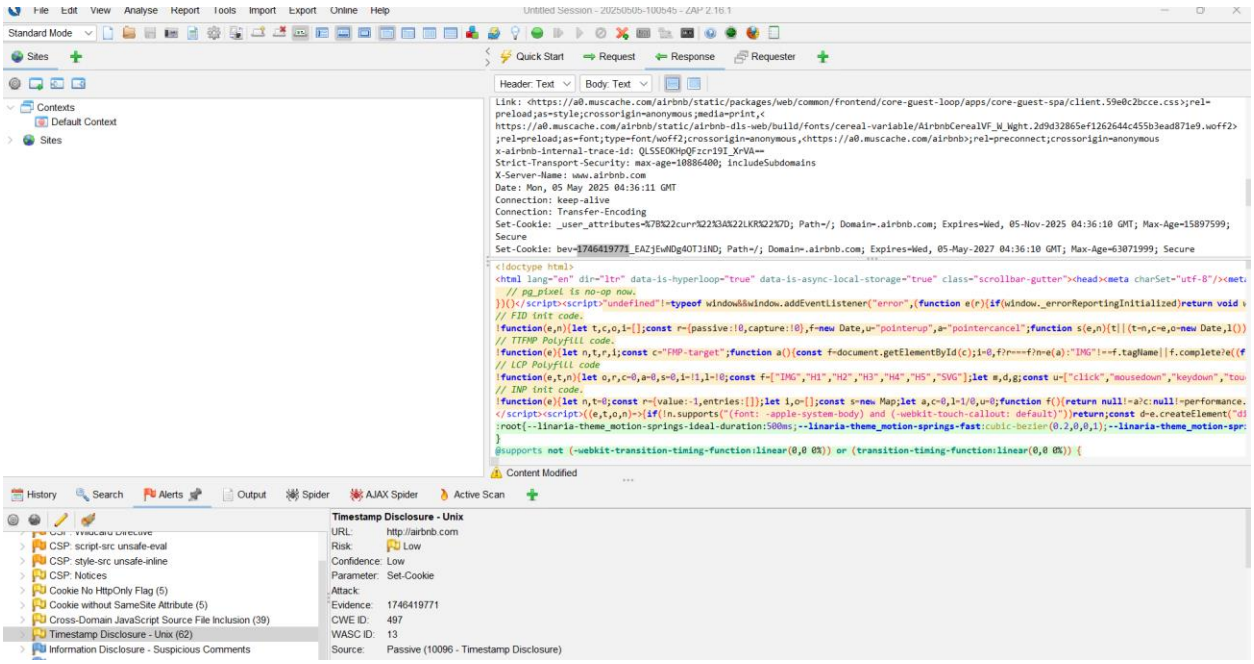
4. **Exploitation & Validation**

Timestamp disclosure Attack Analysis

Timestamp Disclosure is when a web application **reveals Unix timestamps** or other date/time values **in places visible to the client**, like:

- HTTP response headers (e.g., Set-Cookie)
- URLs or query parameters (e.g., ?created=1746419771)
- Hidden form fields
- API responses

.



EXPLOITATION

Step 1: Decode the Timestamp

You find: 1746419771

Use a timestamp converter like:

- https://www.unixtimestamp.com/
- date -d @1746419771 on Linux

Step 2: Analyze How the App Uses the Cookie

Use Burp Suite to:

- Inspect **requests made with this cookie**
- Check if:
    - It appears in **authentication headers**
    - It's used as a **tracker ID**, **session ID**, or **parameter in API calls**

If the cookie is **always unique per session** and time-based, you may have a predictable pattern.

Step 3: Look for Predictable Generation

Say you observe this pattern:

- airbnb_tracker = current_unix_time
- Every new visit gives a cookie like:

Set-Cookie: airbnb_tracker=1746419772

That means the app is generating tracker IDs using:

airbnb_tracker = str(int(time.time()))

That's **predictable**.

Step 4: Write a Script to Generate Timestamps

If the app uses the tracker in URLs or API requests, and the timestamp is predictable, try:

```
import time
import requests
```

```
# Simulate multiple guesses around the known timestamp
for i in range(-10, 10):
    ts = 1746419771 + i
    cookies = {'airbnb_tracker': str(ts)}
    r = requests.get("https://airbnb.com/some_endpoint", cookies=cookies)
    if "Welcome back" in r.text or r.status_code == 200:
        print(f"[+] Possible valid tracker: {ts}")
```

Step 5: Replay or Forge Session (If Possible)

If the app uses this value in session tokens, use the timestamp to forge session or token identifiers.

Example:

```
import hashlib

# Let's assume the session token is hash(userID + timestamp)
user_id = "victim@example.com"
timestamp = "1746419771"
session = hashlib.sha256((user_id + timestamp).encode()).hexdigest()

print("[+] Forged Session ID:", session)
```

Then use Burp to inject it:

```
Cookie: session=FORGED_SESSION
```

Step 6: Combine with Other Vulnerabilities

Now that you have:

- Timestamp disclosure
- Possibly predictable cookies

You could **chain it** with:

- IDOR (to access others' data)
- Weak session handling
- CSRF or login replay

5. **Report Writing**

Title:

Unix Timestamp Disclosure via Set-Cookie Header on http://airbnb.com

Summary:

A passive scan identified a **timestamp disclosure** vulnerability on `http://airbnb.com`. The Set-Cookie header contains a Unix timestamp (`1746419771`) which corresponds to `2025-05-05 10:06:11`. Such disclosures may not be directly exploitable, but they can aid an attacker in fingerprinting server behavior or identifying predictable patterns in session or application data. Timestamp information, especially if included in cookies or headers, can support recon efforts, session prediction, or coordinated timing-based attacks.

Affected Endpoint:

https://airbnb.com

Vulnerability Type:

- Timestamp Disclosure
- Sensitive Information Exposure
- CWE-497: Exposure of System Data
- WASC-13: Information Leakage
- OWASP Top 10:
    - 2021 A01: Broken Access Control *(related through predictable behavior)*
    - 2017 A03: Sensitive Data Exposure

Steps to Reproduce:

Step 1: Access the Site

Visit `http://airbnb.com` or intercept traffic via a proxy (e.g., Burp Suite or OWASP ZAP).

Step 2: Inspect HTTP Response Headers

Examine the HTTP headers returned from the server.

Step 3: Observe the Set-Cookie Header

Step 4: Decode the Timestamp

- Unix timestamp: `1746419771`
- Decoded to: `2025-05-05 10:06:11` (UTC)

Impact:

- Aids in fingerprinting server or application behavior.
- May indicate predictable session or event tracking.
- Could be aggregated with other data to identify patterns.
- Provides recon data that can support advanced timing or session hijacking attacks.

Risk:

- Risk Rating: Low
- Confidence: Low
- Exploitability: Low
- Impact: Low to Moderate (based on context aggregation)

Recommendations:

- Assess whether this timestamp contains sensitive or internal operational data.
- If unnecessary for client-side use, remove or obfuscate the timestamp.
- Implement session or cookie identifiers that do not reveal timestamps or predictable values.
- Ensure cookie values cannot be reverse-engineered into meaningful metadata.

Supporting Evidence:

- Parameter: Set-Cookie
- Attack: Passive

- Evidence: 1746419771
- Decoded Time: 2025-05-05 10:06:11 UTC
- Source: ZAP Passive Scan (Alert 10096 - Timestamp Disclosure)

Additional Notes:

While this timestamp disclosure does not represent a critical issue by itself, it may reveal internal server logic or session generation patterns when combined with other data. It is recommended to review the use of timestamps in client-accessible fields and headers.

References:

- CWE-497: Exposure of System Data(https://cwe.mitre.org/data/definitions/497.html)
- OWASP A01 2021: Broken Access Control(https://owasp.org/Top10/A01_2021-Broken_Access_Control/)
- OWASP A03 2017: Sensitive Data Exposure(https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html)
- CWE-200: Information Exposure(https://cwe.mitre.org/data/definitions/200.html)