Sri Lanka Institute of Information Technology

Year 2 semester 2

IT23360600
G. P. I. Perera

SLIIT KANDY UNI

BUG BOUNTY REPORT 10

Web Security – IE2062

B.Sc. (Hons) in information Technology Specializing in Cyber Security

1. Requirement gathering and analysis

Selected sub domain	support.greenhouse.io
Hakerone URL	https://hackerone.com/greenhouse/
IP address	216.198.54.1, 216.198.53.1

Subdomain list

```
┌──(lynx㉿vbox)-[~]
└─$ subfinder -d support.greenhouse.io

                     projectdiscovery.io

[INF] Current subfinder version v2.6.0 (outdated)
[INF] Loading provider config from /home/lynx/.config/subfinder/provider-config.yaml
[INF] Enumerating subdomains for support.greenhouse.io
www.support.greenhouse.io
[INF] Found 1 subdomains for support.greenhouse.io in 2 seconds 150 milliseconds
```

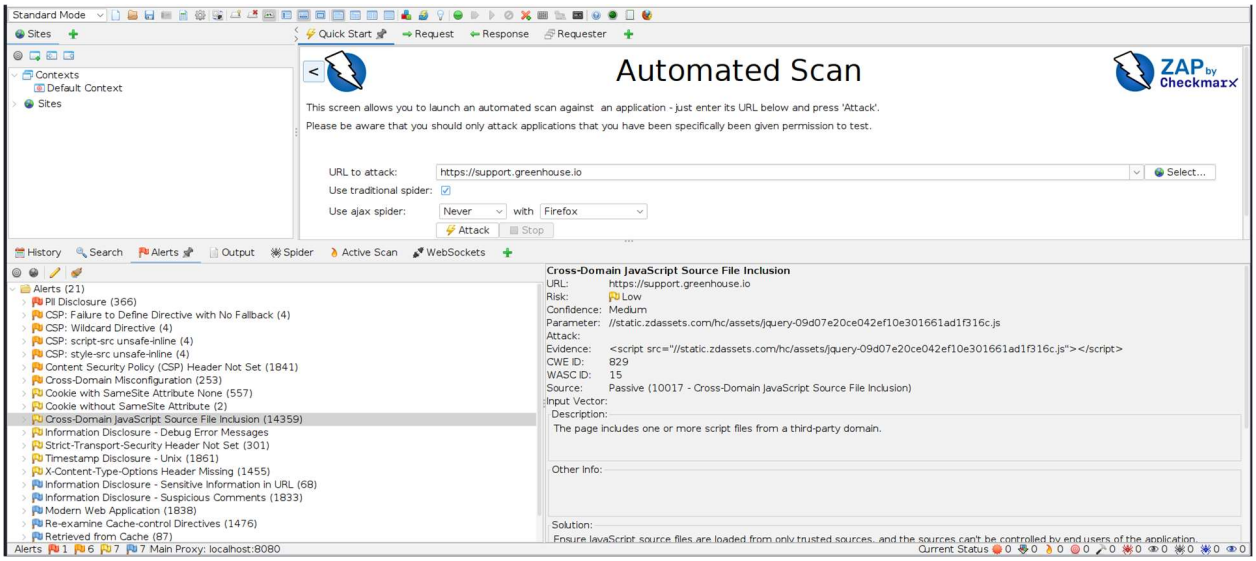
Firewall detection:

```
┌──(lynx㉿vbox)-[~]
└─$ wafw00f https://support.greenhouse.io

      ~ WAFW00F : v2.3.1 ~
    The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://support.greenhouse.io
[+] The site https://support.greenhouse.io is behind Cloudflare (Cloudflare Inc.) WAF.
[~] Number of requests: 2
```

Nmap scan:

```
┌──(lynx㉿vbox)-[~]
└─$ nmap support.greenhouse.io
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-28 00:05 EDT
Nmap scan report for support.greenhouse.io (216.198.54.1)
Host is up (0.22s latency).
Other addresses for support.greenhouse.io (not scanned): 216.198.53.1
Not shown: 995 filtered tcp ports (no-response)
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 34.36 seconds
```

Active scan result from OWASP ZAP:

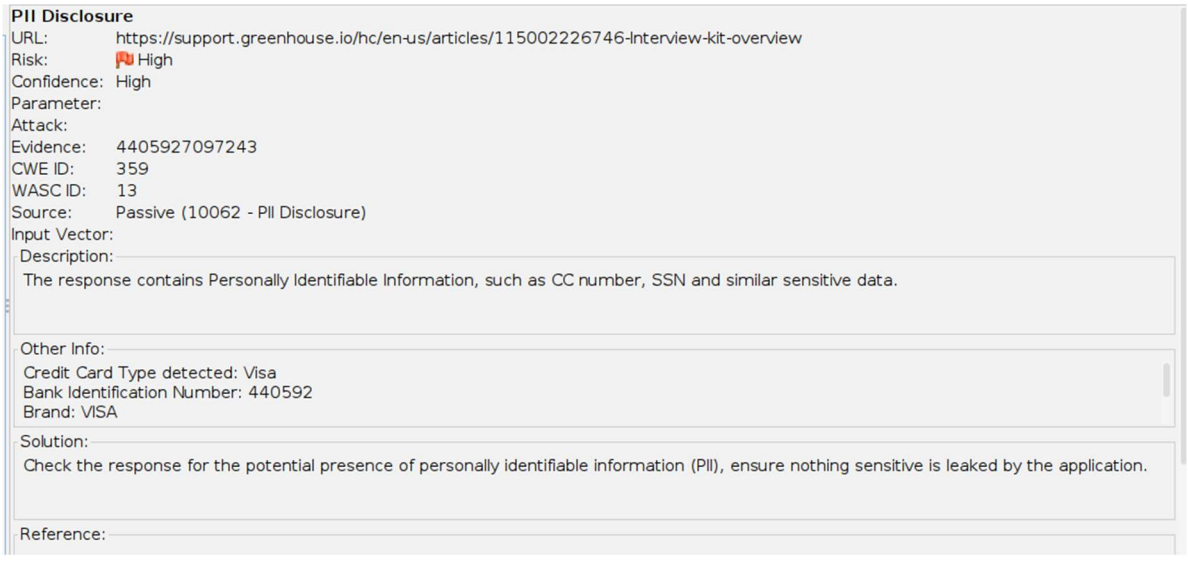


2. Report Details

1. Vulnerability Title – PII disclosure

2. Vulnerability Description:

During security testing, it was observed that sensitive Personally Identifiable Information (PII) - credit card types, Bank identification number, credit card brand and issuer was being exposed through insecure or unauthenticated endpoints.



This type of information was accessible without proper authorization checks, or exposed unintentionally through API responses, web pages, or error messages.

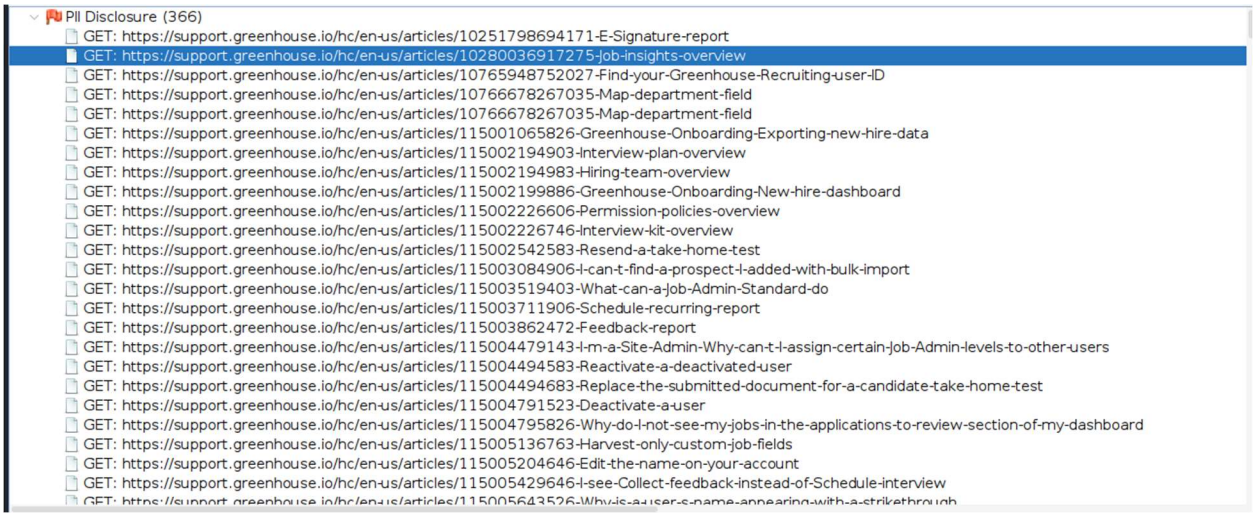
The leak of PII can severely impact user privacy and may lead to further attacks such as:

- **Identity theft**
- **Social engineering attacks**
- **Phishing campaigns**
- **Targeted account takeovers**

In many jurisdictions, mishandling or leaking PII may also result in **legal non-compliance** with data protection regulations like **GDPR**, **CCPA**, or other privacy laws.

3. Affected Components:

- 300+ URLs in https://support.greanhouse.io domain



4. Impact Assessment:

OWASP analysis:

Risk level	High
Confidence	High

5. Steps to reproduce –

- Access the URL and Inspect the HTTP response headers and body for any sensitive information, like identifiers, PII fields, or confidential tokens. Confirm the presence of PII in the response, potentially visible to any user or attacker with network access.
- Look through the web application's pages, focusing on:Forms where users input personal information (registration, checkout, account settings, etc.). URLs that might contain query parameters with personal data (e.g., `user_id=12345`). HTTP response headers (check if PII like session identifiers are included). Cookies that might store PII.
- Check for Exposed Databases or Logs. Sometimes web applications leak sensitive data through database backups, error logs, or misconfigured file systems:
    - Review any publicly accessible files, logs, or backups for inadvertent exposure of PII.

    - Perform a **Directory Traversal** scan to check if files like `.bak`, `.sql`, or other database dumps are exposed.
    - Check for **verbose error messages** that may display database structure or sensitive data.

6. Proposed mitigation or fix

- Apply strict authentication and authorization checks on all endpoints serving user data.
- Minimize PII exposure — only send necessary fields to the frontend.
- Implement proper access control policies (e.g., user A should not access user B's data).
- Monitor and audit APIs for excessive data leakage.
- Comply with relevant data protection regulations (e.g., GDPR, HIPAA, CCPA).