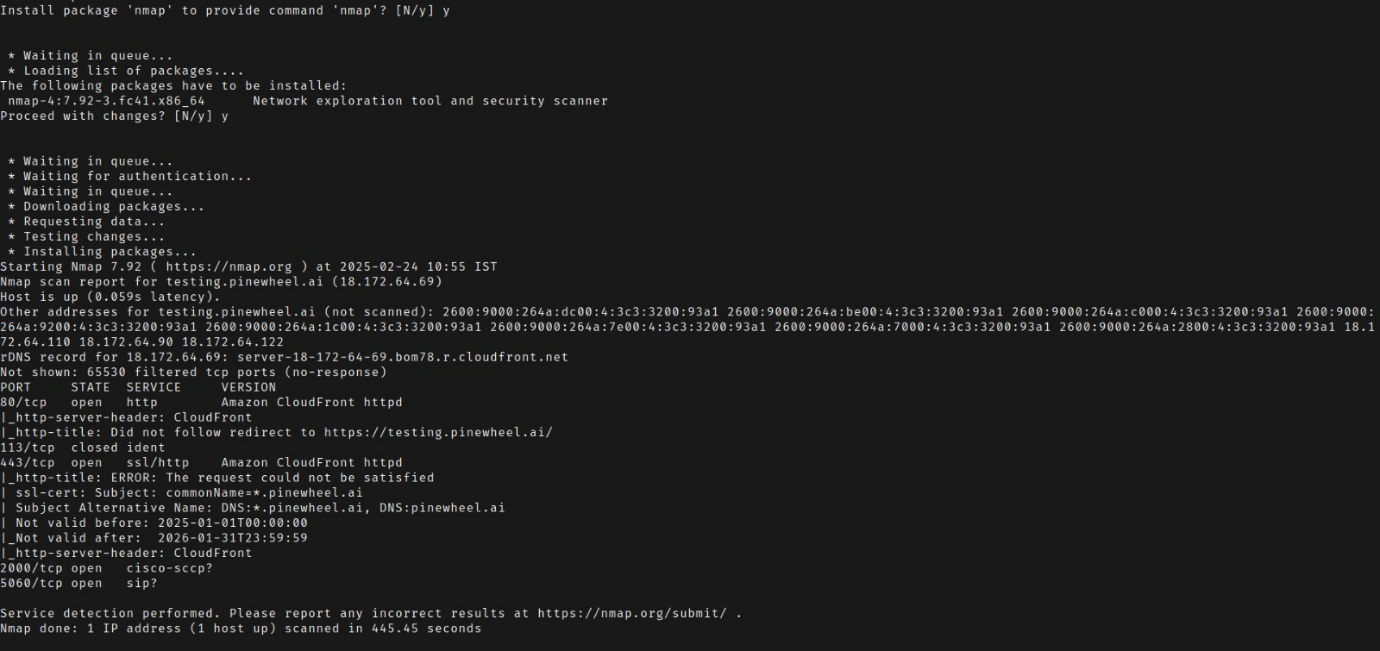
**PineWheel Security Report**

**1. Nmap Scan Results**

**Command:**  
nmap -sC -sV -p- testing.pinewheel.ai

**Output:**

  
Starting Nmap 7.92 ( [https://nmap.org](https://nmap.org/) ) at 2025-02-24 10:55 IST  
Nmap scan report for testing.pinewheel.ai (18.172.64.69)  
Host is up (0.059s latency).

Other addresses for testing.pinewheel.ai (not scanned): 2600:9000:264a:dc00:4:3c3:3200:93a1, 2600:9000:264a:be00:4:3c3:3200:93a1, 2600:9000:264a:c000:4:3c3:3200:93a1, 2600:9000:264a:9200:4:3c3:3200:93a1, 2600:9000:264a:1c00:4:3c3:3200:93a1, 2600:9000:264a:7e00:4:3c3:3200:93a1, 2600:9000:264a:7000:4:3c3:3200:93a1, 2600:9000:264a:2800:4:3c3:3200:93a1.

**rDNS Record:**  
server-18-172-64-69.bom78.r.cloudfront.net

**Open Ports:**

* **80/tcp:** CloudFront HTTP Server
  + *http-server-header:* CloudFront
  + *http-title:* Redirects to HTTPS
* **113/tcp:** Closed Ident
* **443/tcp:** Open SSL/HTTP
  + *http-title:* ERROR: The request could not be satisfied
  + *SSL Cert:*
    - Common Name: \*.pinewheel.ai
    - Subject Alternative Name: DNS: \*.pinewheel.ai, DNS: pinewheel.ai
    - Validity: 2025-01-01 to 2026-01-31
* **2000/tcp:** Open
* **5060/tcp:** Open

**Reason for Scan:**

* Identifies all open ports on the target.
* Detects services running on those ports (e.g., web servers, SSH, FTP, databases).
* Tries to determine the exact versions of the services.
* Runs default scripts to collect additional details like HTTP headers, SSL certificate info, or known vulnerabilities.

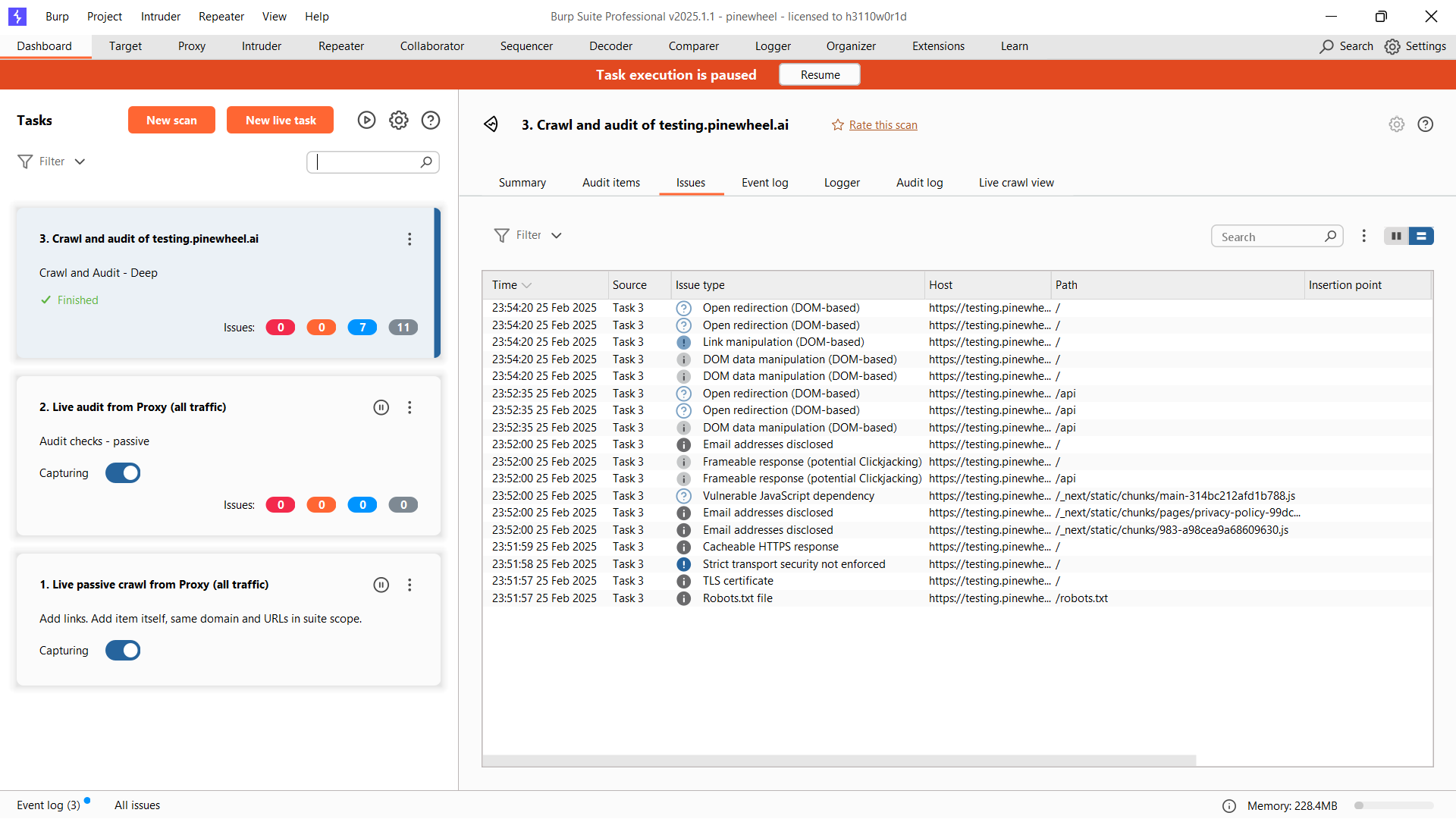
**2. SQL Injection Test**

**Command:**  
sqlmap -u "<https://testing.pinewheel.ai/index.php?id=1>" --dbs --level=5 --risk=3 --random-agent

**Output:**

* Parameter 'User-Agent' appears to be 'SAP MaxDB stacked queries (heavy query)' injectable.
* Backend DBMS detected as **SAP MaxDB**.
* Warning: Possible false positive detected. Fingerprinting also indicates **PostgreSQL**.
* Parameter 'Host' appears to be vulnerable to **AND boolean-based blind SQL injection**.

**3.Burp Suite Findings**



**3.1 DOM-Based Link Manipulation Vulnerability**

* **Severity:** Low
* **Confidence:** Firm
* **Affected URL:** <https://testing.pinewheel.ai/>
* **Issue:**
  + The application reads data from location.href and assigns it to an element’s href attribute dynamically.
  + This allows attackers to manipulate links and redirect users to malicious sites.

**Potential Exploits:**

* **Phishing Attacks:** Trick users into clicking manipulated links that redirect them to malicious sites.
* **Form Data Theft:** Modify form submission URLs to steal user data.
* **Unintended Actions:** Change the target of links to perform unauthorized actions.
* **Bypassing XSS Defenses:** Inject links that execute malicious scripts.

**3.2 DOM-Based Data Manipulation Vulnerability**

* **Severity:** Informational
* **Confidence:** Firm
* **Affected URL:** <https://testing.pinewheel.ai/api>
* **Issue:**
  + The application reads data from location.href and passes it to history.replaceState.
  + This may allow attackers to manipulate the UI or mislead users.

**Potential Exploits:**

* **Virtual Defacement:** Attackers modify the webpage appearance to mislead users.
* **Inducing Unintended Actions:** Users unknowingly perform unauthorized actions.
* **Session Tampering:** Attackers modify URL parameters to persist altered data in browser history.

**3.3 Clickjacking Vulnerability**

* **Severity:** Informational
* **Confidence:** Firm
* **Affected URL:** <https://testing.pinewheel.ai/api>
* **Issue:**
  + The application does not set **X-Frame-Options** or **Content-Security-Policy (CSP)** headers.
  + This makes it possible to load the page within an <iframe>.

**Potential Exploits:**

* **Clickjacking Attack:**
  + The website is embedded in an invisible <iframe>.
  + Fake buttons or forms overlay the real site, leading to unintentional actions.
* **Bypassing CSRF Protections:** Clickjacking can bypass CSRF protections by tricking users into unintended requests.
* **Credential Theft & Phishing:** Attackers may capture user keystrokes or trick users into entering sensitive data into fake forms.

**Conclusion & Recommendations:**

* **Fix SQL Injection Risks:** Implement input validation and prepared statements.
* **Mitigate DOM Manipulation Issues:** Avoid using location.href for setting element attributes.
* **Prevent Clickjacking:** Implement X-Frame-Options and CSP headers.
* **Improve Security Headers:** Set proper Content Security Policy rules to mitigate potential attacks.
* **Regular Security Audits:** Perform routine penetration testing to detect vulnerabilities early.

**End of Report**