Sri Lanka Institute of Information Technology

BUG BOUNTY REPORT 10

(Ubiquiti Inc. Web site)

IE2062 – Web Security

W.A.K.S Wijethunga

IT23361768

Table of Contents

1. Introduction to bug bounty program and audit scope

❖ Ubiquiti Inc.

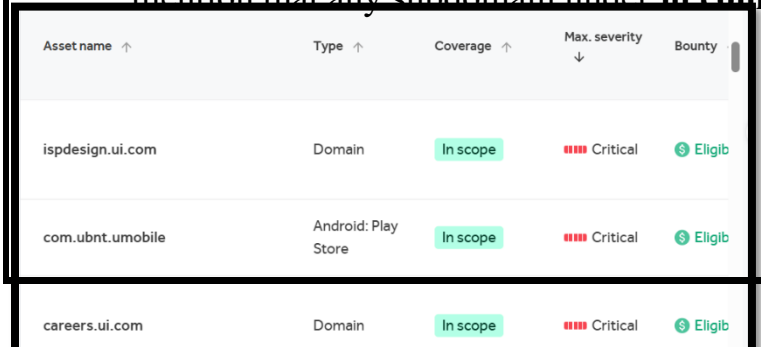
Ubiquiti Inc. is a global technology company that develops and manufactures networking products for service providers, enterprises, and home users. Known for its popular product lines such as UniFi, AmpliFi, and EdgeMAX, Ubiquiti delivers advanced networking solutions including Wi-Fi systems, security cameras, routers, and IoT devices. These products are managed through web-based platforms and cloud services such as ui.com and unifi.ui.com, which are critical for device configuration, monitoring, and user management.

Due to the nature of these platforms—handling sensitive user data, device credentials, and network configurations—web application security is a top priority for Ubiquiti. This report documents a vulnerability identified during a responsible security assessment, in alignment with ethical disclosure standards and bug bounty guidelines. The objective is to assist Ubiquiti in enhancing the security posture of its web infrastructure and protecting its global user base from potential exploitation.

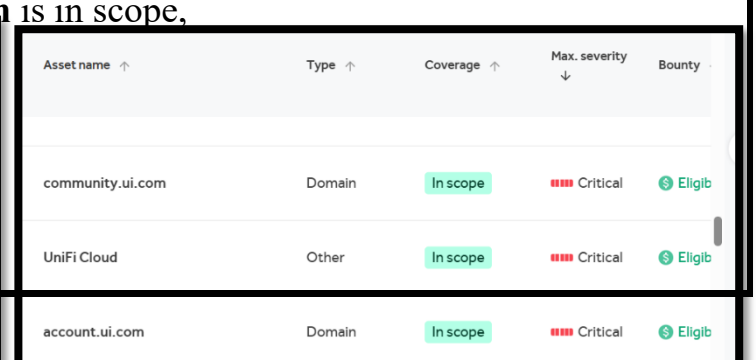
In Hackerone bug bounty program, they defined these subdomains (and all inclusive) as valid subdomains for testing.

- **ispdesign.ui.com**
- **ispdesign.ui.com**
- **careers.ui.com**
- **ir.ui.com**
- **fw-update.ubnt.com**
- **unifi.ui.com**
- **uisp.com**
- **rma.ui.com**
- **community.ui.com**
- **account.ui.com**

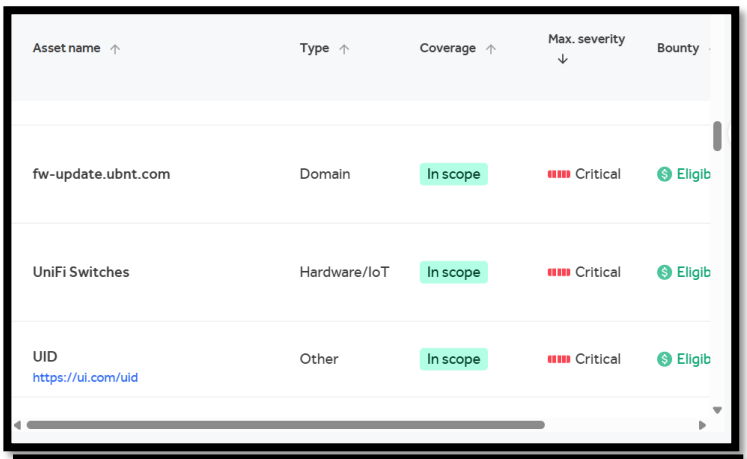
Eligible in-scope subdomains for bug bounty program are mentioned below and they mention that any subdomain under **ui.com** is in scope,



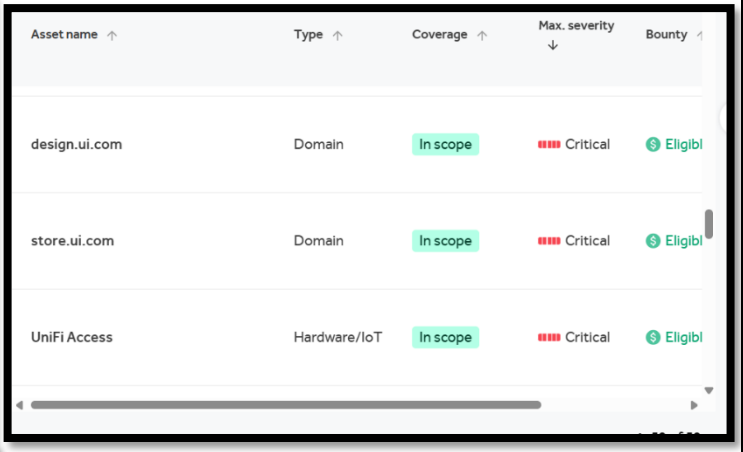
Asset name	Type	Coverage	Max. severity	Bounty
ispdesign.ui.com	Domain	In scope	Critical	Eligib
com.ubnt.umobile	Android: Play Store	In scope	Critical	Eligib
careers.ui.com	Domain	In scope	Critical	Eligib



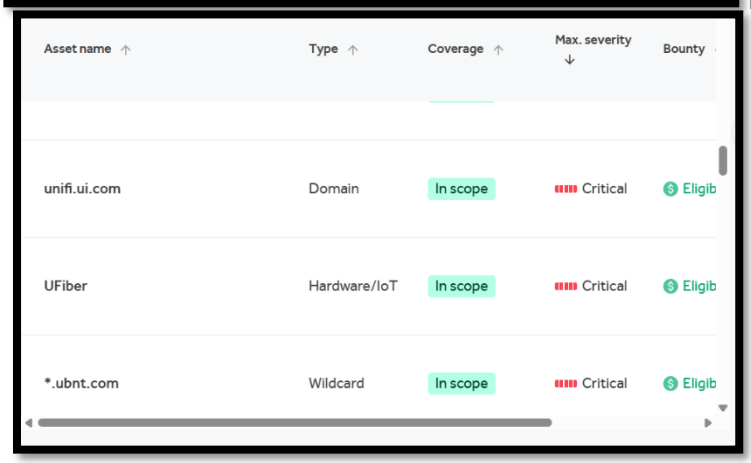
Asset name	Type	Coverage	Max. severity	Bounty
community.ui.com	Domain	In scope	Critical	Eligib
UniFi Cloud	Other	In scope	Critical	Eligib
account.ui.com	Domain	In scope	Critical	Eligib



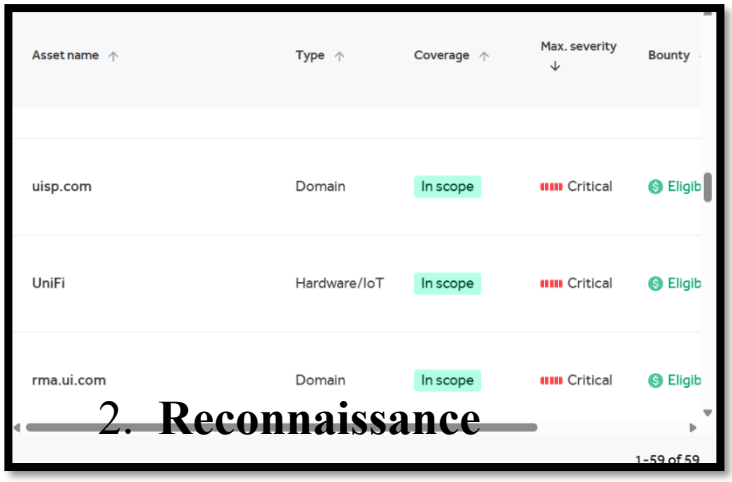
Asset name	Type	Coverage	Max. severity	Bounty
fw-update.ubnt.com	Domain	In scope	Critical	Eligib
UniFi Switches	Hardware/IoT	In scope	Critical	Eligib
UID https://ui.com/uid	Other	In scope	Critical	Eligib



Asset name	Type	Coverage	Max. severity	Bounty
design.ui.com	Domain	In scope	Critical	Eligibl
store.ui.com	Domain	In scope	Critical	Eligibl
UniFi Access	Hardware/IoT	In scope	Critical	Eligibl



Asset name	Type	Coverage	Max. severity	Bounty
unifi.ui.com	Domain	In scope	Critical	Eligib
UFiber	Hardware/IoT	In scope	Critical	Eligib
*.ubnt.com	Wildcard	In scope	Critical	Eligib



Asset name	Type	Coverage	Max. severity	Bounty
uisp.com	Domain	In scope	Critical	Eligib
UniFi	Hardware/IoT	In scope	Critical	Eligib
rma.ui.com	Domain	In scope	Critical	Eligib

1-59 of 59

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,

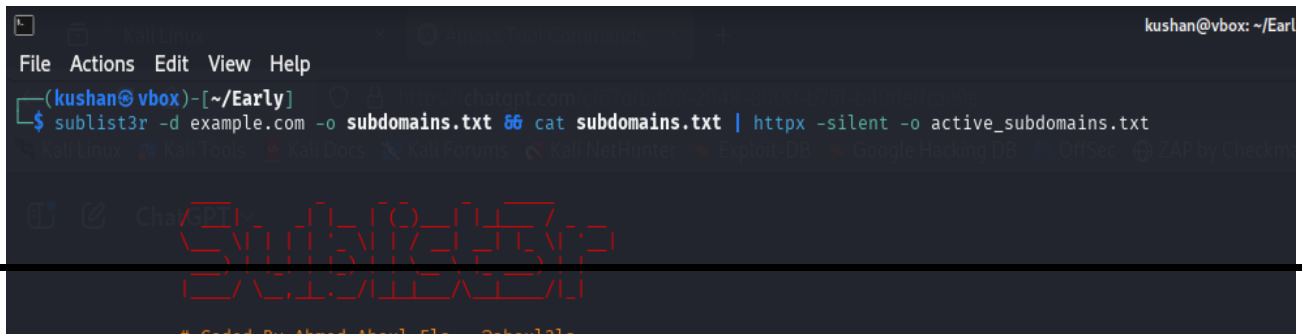
sudo pip install -r requirements.txt

After installing the requirements, enter
sublist3r -d ui.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.

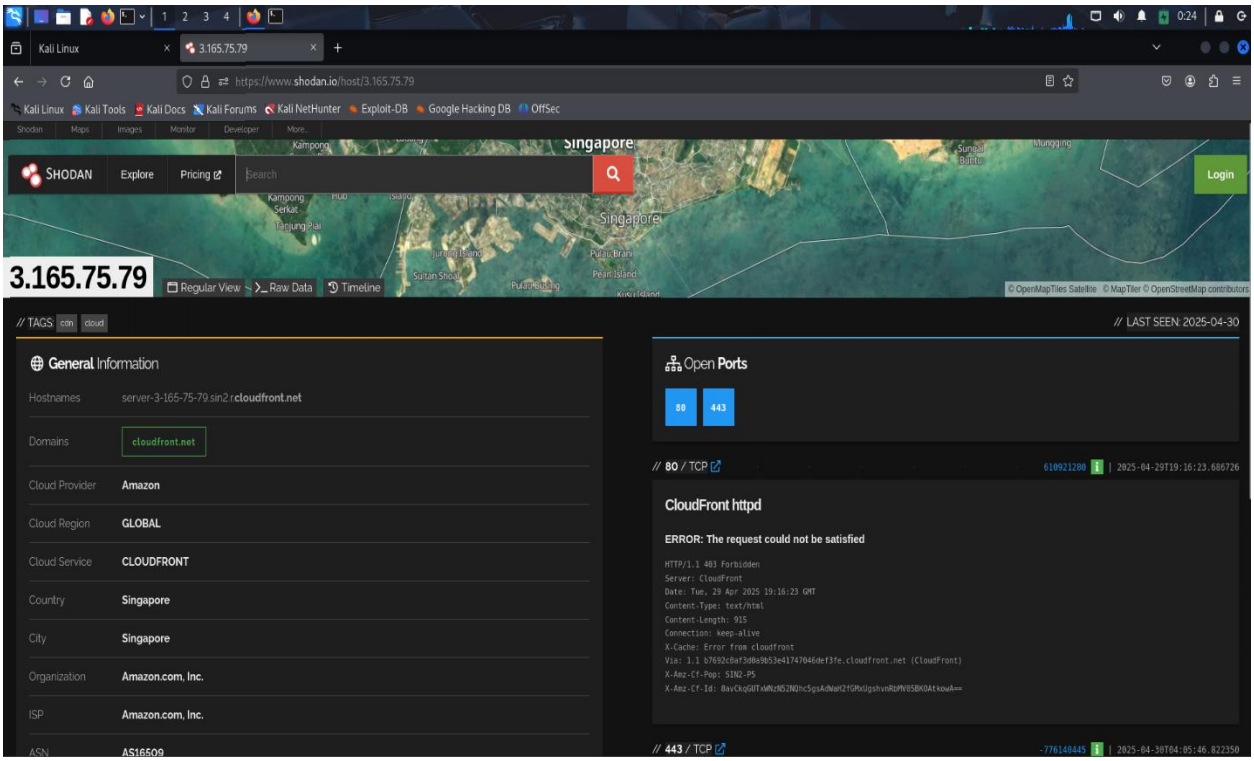
Following the completion of the scan, the findings reveal that the majority of the subdomains are indeed active.



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

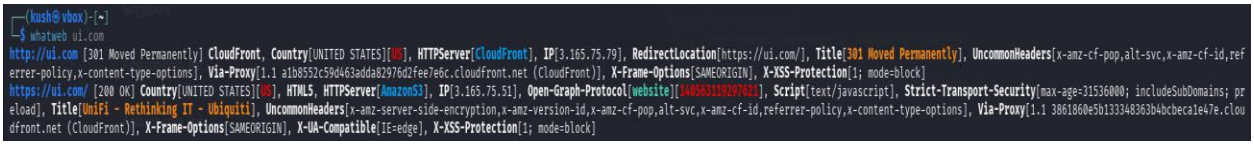
Whatweb is a powerful open-source tool designed to identify the technologies used by websites. It works by analyzing the responses from a web server, such as

HTTP headers, HTML content, cookies, and scripts, to detect the underlying technologies.

To detect technologies used by a website, simply run :

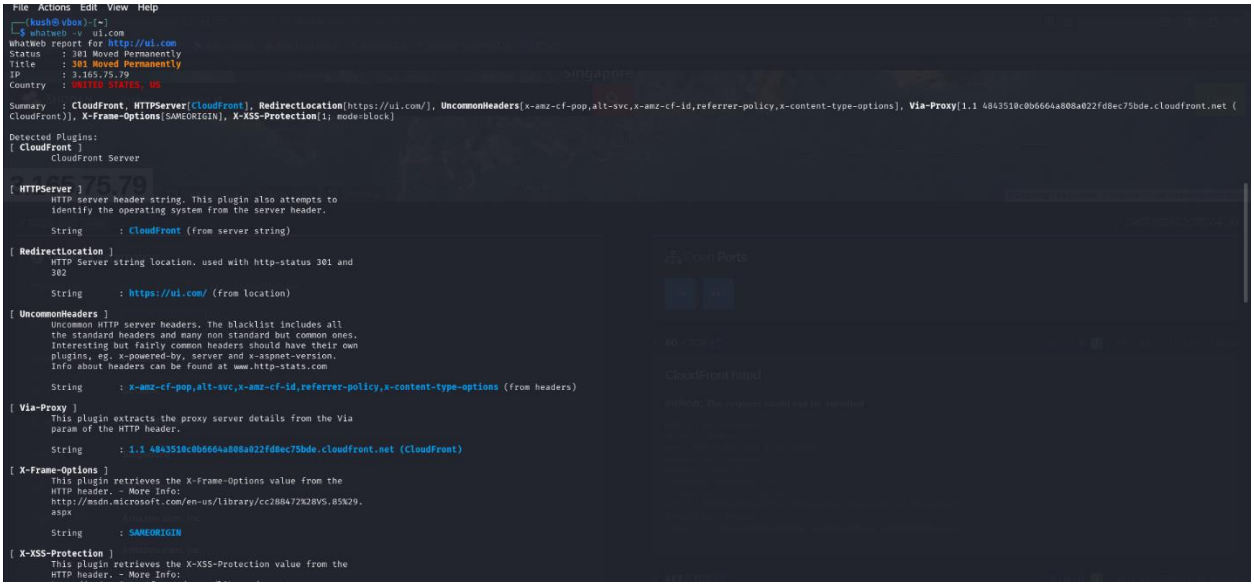
whatweb ui.com

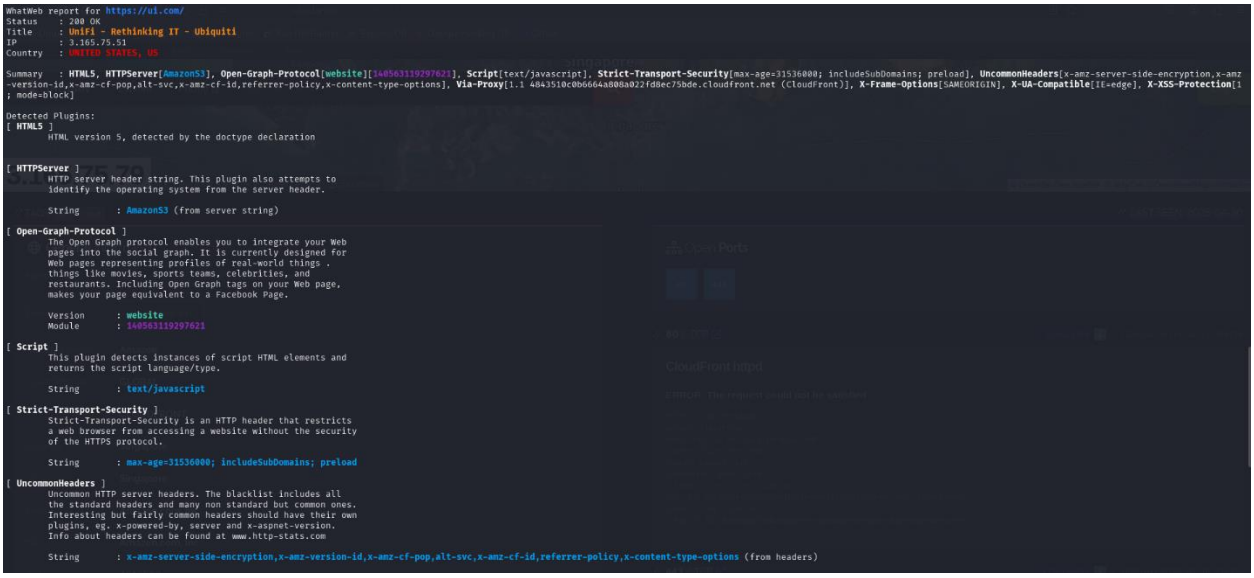
This command will analyze the website and display a summary of the detected technologies**.**

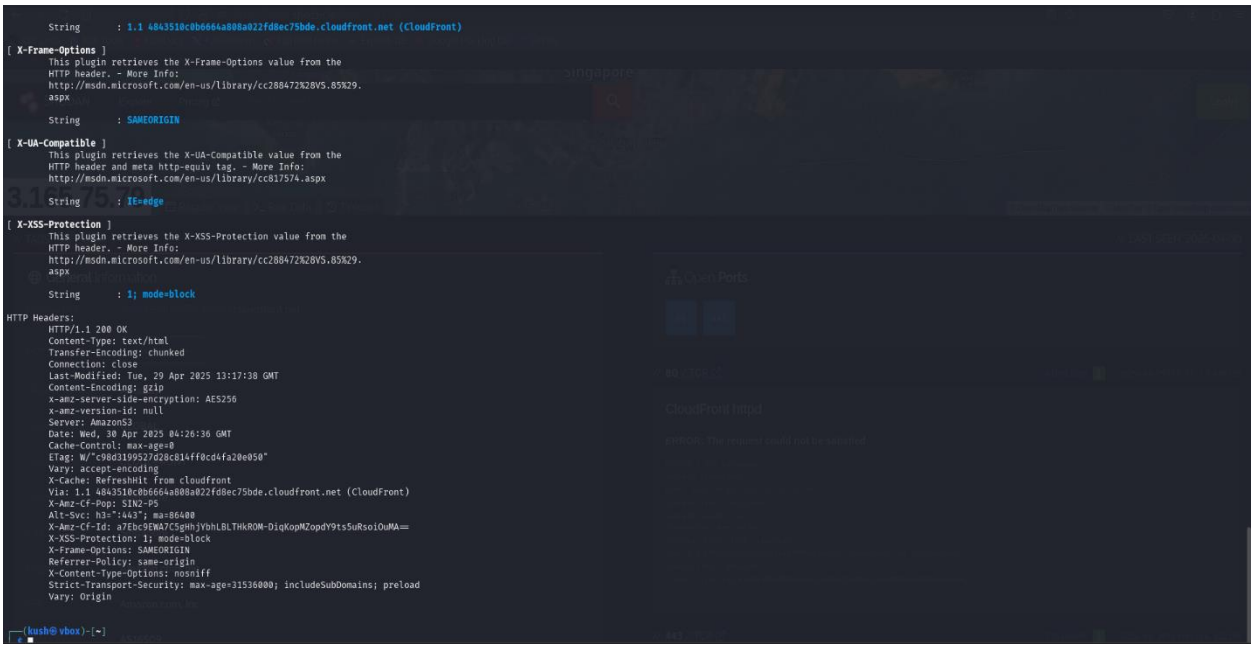


To get detailed information about the detection process:

whatweb -v ui.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 **ui.com,**

Identify services running on open ports,

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

ii. Web vulnerabilities

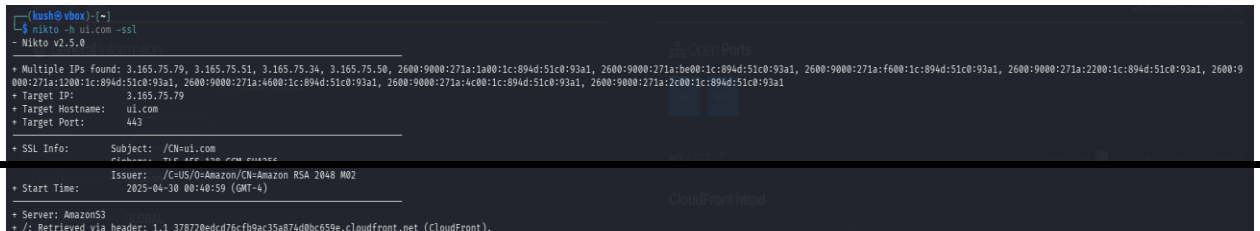
Nikto is an open-source web server scanner designed to identify vulnerabilities, outdated software, and security misconfigurations on web servers. It performs comprehensive testing for over 6700

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

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

Scans both HTTP and HTTPS,

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



Automated Testing

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

OWASP ZAP

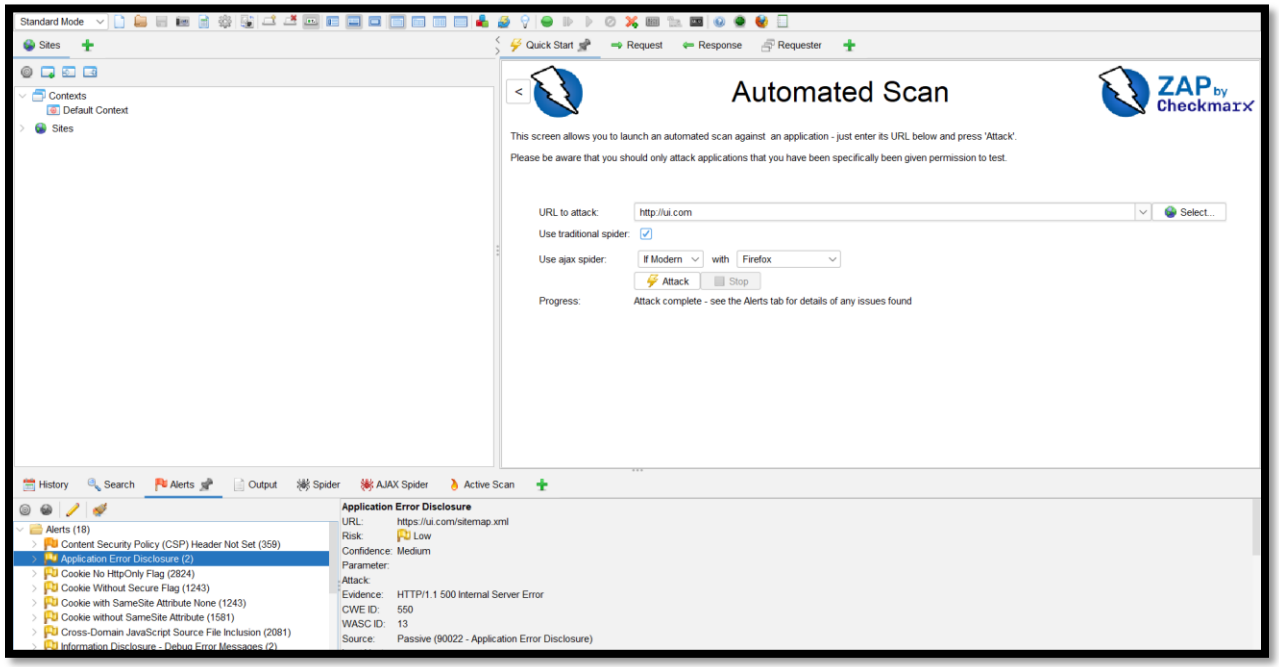
The Open Web Application Security Project Zed Attack Proxy (OWASP ZAP) is an open-source vulnerability scanner renowned for its capability to function as a Man-in-the-Middle (MITM) proxy. It assesses various vulnerabilities by scrutinizing responses from the web application or server. Notably convenient 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. Access-Control-Allow-Origin: *

Risk: The presence of a wildcard (*) in the Access-Control-Allow-Origin header means any external origin is allowed to access the website's resources via CORS (Cross-Origin Resource Sharing).

Impact:

- Malicious websites can make authenticated or unauthenticated API calls to the target domain using JavaScript.
- If any API response contains sensitive data (e.g., user profile info, emails, internal IDs), this information could be leaked to an attacker's domain.
- Could allow attackers to bypass same-origin policies and steal user data, especially if Access-Control-Allow-Credentials is also present or misconfigured.
- In combination with cookies or localStorage/sessionStorage values, attackers may perform session hijacking or privilege escalation.

2. Missing X-Content-Type-Options Header

Risk: The absence of the X-Content-Type-Options: nosniff header allows browsers to guess (or "sniff") the MIME type of a file.

Impact:

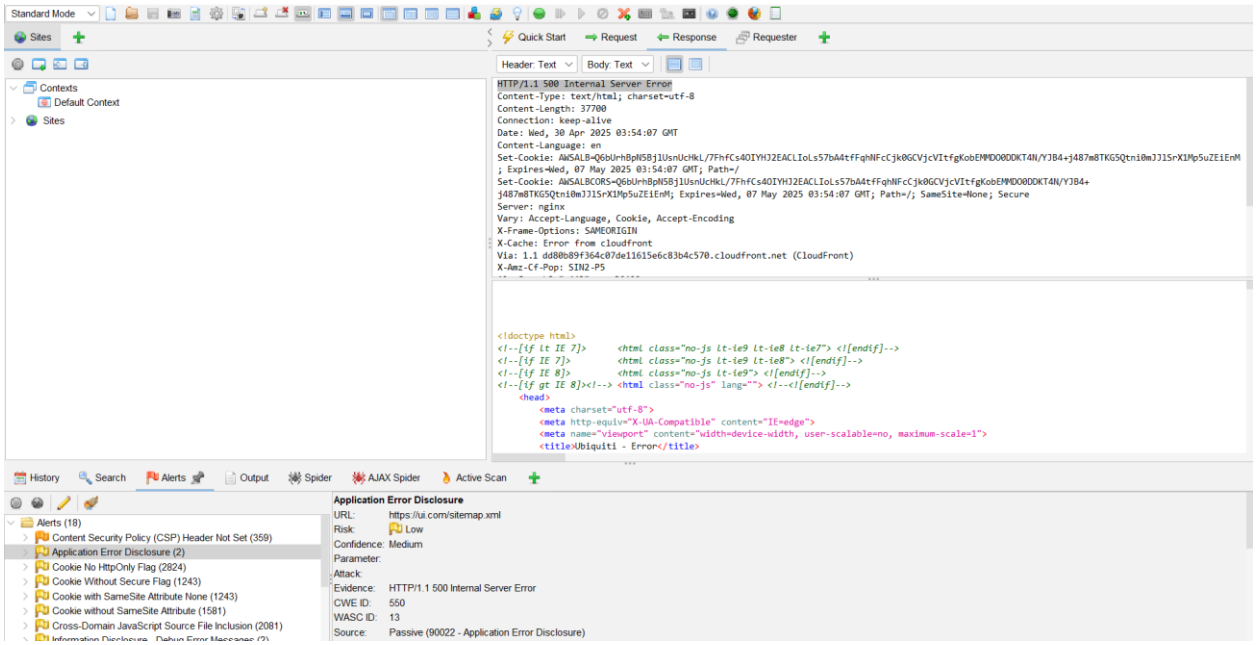
- An attacker could trick the browser into executing a file (e.g., HTML, JavaScript) as a different content type than intended.
- Combined with file upload functionality, this could allow stored XSS or arbitrary code execution in the browser context.
- It increases the chances of MIME-type confusion attacks, especially when serving user-uploaded or untrusted content.
- For example, an uploaded file meant to be downloaded as plain text might be rendered/executed as JavaScript if the MIME sniffing is allowed.

4. **Exploitation & Validation**

Application Error Disclosure Attack Analysis

During passive security testing of the target application, an **Application Error Disclosure** vulnerability was discovered on the endpoint https://ui.com/sitemap.xml. This vulnerability occurs when a web application fails to handle server-side errors securely and instead exposes internal system details—such as stack traces, file paths, or configuration data—to the client in the HTTP response. While often overlooked, these disclosures provide valuable insight to an attacker, aiding in reconnaissance and making it easier to craft targeted attacks.

The presence of a 500 Internal Server Error indicates an unhandled exception on the server side. If error details are returned to the user, they can reveal information about the backend architecture, technologies used, and even coding errors—putting the application's security posture at risk. This class of vulnerability falls under OWASP Top 10 – A05: Security Misconfiguration and CWE-550: Exposure of Sensitive Information to an Unauthorized Actor.



How to Exploit Application Error Disclosure:

1. Trigger the Error Repeatedly

- The attacker visits or sends automated requests to https://ui.com/sitemap.xml (or similar endpoints).
- The server consistently responds with 500 Internal Server Error.

2. Inspect the Response Content

- The attacker checks if the HTTP response body includes **debug information** such as:
    - Full or partial **stack trace**
    - **File paths** (e.g., /var/www/ui/api/sitemap_handler.py)
    - **Error messages** (e.g., NullReferenceException, NoMethodError, SQLSTATE)
    - **Technology/framework info** (e.g., Django, PHP, Spring, etc.)

🔎 Tools like Burp Suite, OWASP ZAP, curl, or browser DevTools can help inspect the raw HTTP response.

3. Leverage the Information for Further Attacks

- **File Paths**: Useful for LFI/RFI (Local/Remote File Inclusion), or navigating directory traversal paths.
- **Stack Traces**: Reveal function names, modules, libraries, or code that may have known CVEs.
- **Technology Disclosure**:
    - If it shows PHP 7.1.33, the attacker searches for known vulnerabilities in that version.
    - If it shows MySQL, attacker may try SQL Injection payloads compatible with MySQL.

4. Enumerate More Endpoints

- Once one error page is found, an attacker may test other URLs to see if similar stack traces appear.
- URLs like:
    - /api/
    - /sitemap.xml

- /debug
- /search?q=test
- /admin/config

5. Automated Enumeration

- Tools like **dirsearch**, **ffuf**, or **gobuster** can be used to scan for more error-prone endpoints.
- Responses with status codes like 500, 502, or 503 are flagged for manual inspection.

5. **Report Writing**

Title:

Application Error Disclosure on https://ui.com/sitemap.xml Reveals Internal Server Error and Potentially Sensitive Information

Summary:

A misconfiguration was identified on https://ui.com/sitemap.xml, where the endpoint returns a 500 Internal Server Error, revealing server-side stack traces or error-handling issues. This behavior indicates improper error handling mechanisms, which may expose sensitive system details such as internal file paths, frameworks in use, or environment-specific configurations. These disclosures can aid attackers in crafting more precise and effective attacks against the web application.

Affected Endpoint:

https://ui.com/sitemap.xml

Vulnerability Type:

- **Security Misconfiguration**
- **Improper Error Handling**
- **CWE-550: Exposure of Sensitive Information to an Unauthorized Actor**
- **WASC-13: Information Leakage**
- **OWASP Top 10:**
    - 2021 A05: Security Misconfiguration
    - 2017 A06: Security Misconfiguration

Steps to Reproduce:

Step 1: Direct Access

- Navigate to the URL: https://ui.com/sitemap.xml in a browser or proxy tool.

Step 2: Observe the Response

- The server returns the following HTTP response:
- HTTP/1.1 500 Internal Server Error
- Depending on configuration, additional error message content may be present in the response body or headers.

Step 3: Check for Disclosure

- Inspect the body of the response or logs for:
    - File paths
    - Stack traces
    - Error messages revealing technology or structure

Impact:

- **Reconnaissance Aid:** Attackers can gather sensitive details (e.g., technology stack, file paths, misconfigured modules).
- **Targeted Exploits:** Leaked internal information could help attackers identify vulnerable entry points or craft more targeted payloads.
- **Application Integrity:** Repeated unhandled errors can indicate deeper architectural or security gaps.

Risk:

- **Risk Rating:** Medium
- **Confidence:** Medium
- **Exploitability:** Low Complexity (Passive)
- **Impact:** Moderate (Information Disclosure)

Recommendations:

- **Implement Custom Error Pages:** Ensure the application does not expose raw error messages or stack traces to end-users.

- **Centralize Logging:** Log detailed error messages securely on the server side, and provide users with generic messages and reference IDs.
- **Code Review:** Audit the endpoint and surrounding code for unhandled exceptions.
- **Security Testing:** Run automated and manual error-handling tests across all public-facing endpoints.

Supporting Evidence:

- **Response Code:** HTTP/1.1 500 Internal Server Error
- **Parameter:** Not applicable (static resource)
- **Attack:** Passive (no payloads required)
- **Source:** ZAP Passive Scanner Alert 90022 – Application Error Disclosure

Additional Notes:

While no stack trace was visible in this instance, the mere presence of a 500 error for a sitemap endpoint suggests an unhandled exception or backend misconfiguration. If sensitive error messages are returned in other environments (e.g., staging or debug mode), they could significantly increase the application's attack surface.

References:

- CWE-550: Exposure of Sensitive Information
- OWASP Top 10: A05 - Security Misconfiguration (2021)
- OWASP WSTG-ERRH-01: Improper Error Handling