Table of Contents

[Conduct a Penetration Test 1](#_Toc191760843)

[**Detect Web Server Version** 2](#_Toc191760844)

[**Target Website:** 2](#_Toc191760845)

[**Detect Open Ports:** 3](#_Toc191760846)

[**Using Nikto:** 3](#_Toc191760847)

[**Vulnerability Assessment Report** 5](#_Toc191760848)

[Target: www.hackthissite.org 5](#_Toc191760849)

[Date: 2025-03-01 22:23:09 (GMT-5) 5](#_Toc191760850)

[Scanner Used: OWASP ZAP / Nikto 5](#_Toc191760851)

[**1. Summary of Findings** 5](#_Toc191760852)

[**2. Detailed Findings & Recommendations** 5](#_Toc191760853)

[1️ Clickjacking Attack Possible (No X-Frame-Options Header) 5](#_Toc191760854)

[2️ BREACH Attack Possible (Content-Encoding: deflate) 6](#_Toc191760855)

[3️ Cookies Missing Secure & HttpOnly Flags 6](#_Toc191760856)

[4️ No X-Content-Type-Options Header (MIME Sniffing Risk) 6](#_Toc191760857)

[5️ Access-Control-Allow-Origin: \*\*\* (CORS Risk) 6](#_Toc191760858)

[6️ robots.txt Exposes /missions/ Directory 7](#_Toc191760859)

[7️ Hostname Mismatch in SSL Certificate 7](#_Toc191760860)

[**3. Risk Assessment Summary** 7](#_Toc191760861)

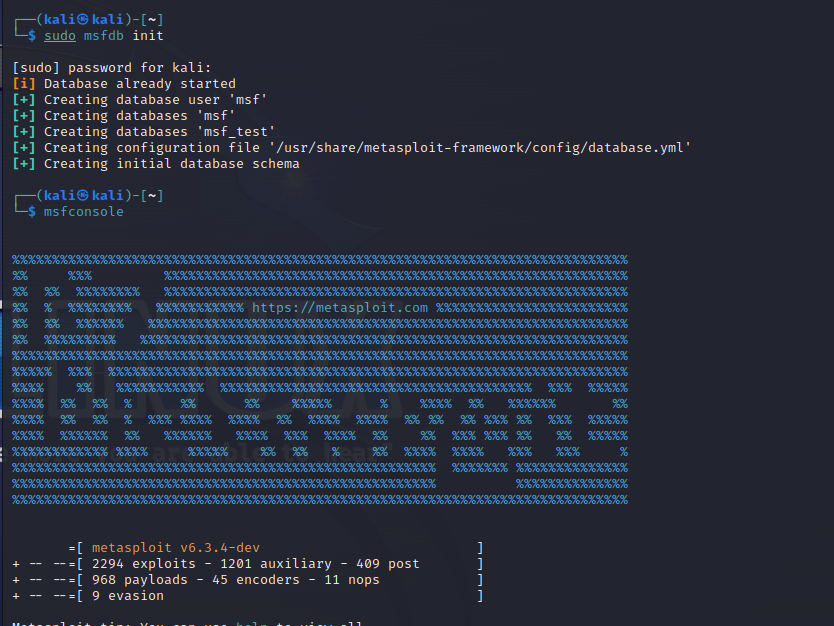
[**4. Conclusion & Next Steps** 7](#_Toc191760862)

Task 2:

# Conduct a Penetration Test

* Perform a basic penetration test on a web application.
* Use tools like Burp Suite or Metasploit to find security vulnerabilities.
* Document findings and suggest remediation steps.

Lets start **msfconsole**

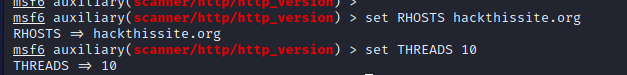


Database attached and msfconsole started

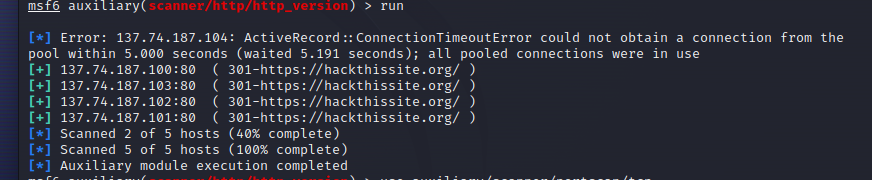
**Detect Web Server Version**

## **Target Website:**

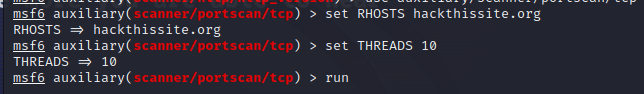
[**https://www.hackthissite.org/**](https://www.hackthissite.org/)

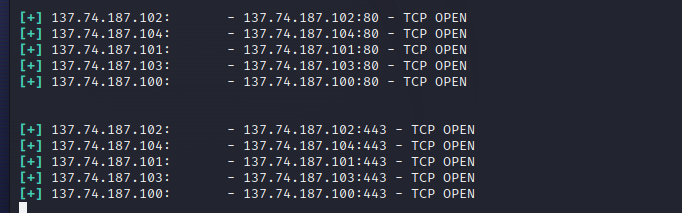
****

**Using 10 threads**

****

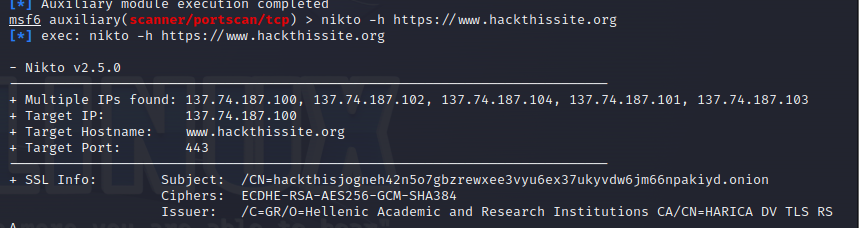
## **Detect Open Ports:**

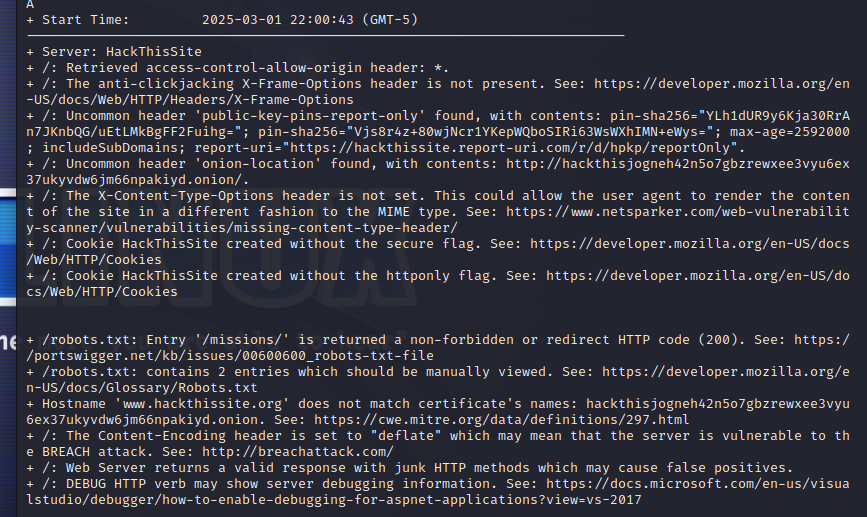
****

****

## **Using Nikto:**

**Using nikto to find known vulnerabilities in website**

****

****

# **Vulnerability Assessment Report**

### **Target:** [www.hackthissite.org](http://www.hackthissite.org/)

### **Date:** 2025-03-01 22:23:09 (GMT-5)

### **Scanner Used:** OWASP ZAP / Nikto

## **1. Summary of Findings**

This report summarizes the vulnerabilities found on www.hackthissite.org. The identified issues may expose the website to security threats such as **clickjacking, cookie hijacking, MIME sniffing attacks, and the BREACH attack**.

|  |  |  |
| --- | --- | --- |
| **Risk Level** | **Vulnerability** | **Impact** |
| **High** | **No X-Frame-Options Header** | Allows Clickjacking Attacks |
| **High** | **BREACH Attack Possible (Content-Encoding: deflate)** | Can lead to sensitive data leakage |
| **Medium** | **Cookies Missing Secure & HttpOnly Flags** | Increases risk of session hijacking |
| **Medium** | **No X-Content-Type-Options Header** | Allows MIME-sniffing attacks |
| **Low** | **Access-Control-Allow-Origin** | Risk of Cross-Origin Resource Sharing (CORS) abuse |
| **Low** | **robots.txt exposes /missions/** | May expose sensitive directories |
| **Low** | **Public Key Pinning Report Only Mode** | Misconfiguration in certificate validation |

## **2. Detailed Findings & Recommendations**

### **1️ Clickjacking Attack Possible (No X-Frame-Options Header)**

* **Description:** The X-Frame-Options header is missing. This allows attackers to embed the website in an **iframe** and trick users into performing unintended actions.
* **Impact:** Attackers can use **UI Redressing** to steal sensitive user interactions.
* **Fix:**
  + Add the following HTTP header:
  + X-Frame-Options: SAMEORIGIN
  + Or use **Content Security Policy (CSP)**:
  + Content-Security-Policy: frame-ancestors 'none';
* **Reference:** MDN: X-Frame-Options

### **2️ BREACH Attack Possible (Content-Encoding: deflate)**

* **Description:** The Content-Encoding header is set to "deflate", which may allow the **BREACH attack** to extract sensitive data from compressed responses.
* **Impact:** Attackers can recover CSRF tokens, session IDs, or passwords.
* **Fix:**
  + Disable **HTTP compression** for sensitive responses.
  + Implement **CSRF tokens with random padding**.
* **Reference:** BREACH Attack

### **3️ Cookies Missing Secure & HttpOnly Flags**

* **Description:** The HackThisSite cookie is missing **Secure** and **HttpOnly** flags.
* **Impact:**
  + **Without Secure flag**: Cookies can be **stolen over HTTP** via MITM attacks.
  + **Without HttpOnly flag**: Cookies are accessible via JavaScript, making them vulnerable to **XSS attacks**.
* **Fix:**
  + Set cookie attributes in the server response:
  + Set-Cookie: HackThisSite=abcd1234; Secure; HttpOnly; SameSite=Strict
* **Reference:** MDN: Secure and HttpOnly Cookies

### **4️ No X-Content-Type-Options Header (MIME Sniffing Risk)**

* **Description:** The X-Content-Type-Options header is missing, which allows **MIME-type sniffing attacks**.
* **Impact:** Attackers can force browsers to **execute malicious scripts** as different content types.
* **Fix:**
  + Add this HTTP header:
  + X-Content-Type-Options: nosniff
* **Reference:** Netsparker: MIME Sniffing Vulnerabilities

### **5️ Access-Control-Allow-Origin: \*\*\* (CORS Risk)**

* **Description:** The Access-Control-Allow-Origin header is set to \*, allowing any website to make requests.
* **Impact:** **Cross-Origin Resource Sharing (CORS) abuse** can occur, leading to **data theft**.
* **Fix:**
  + Restrict CORS to trusted domains:
  + Access-Control-Allow-Origin: https://trusted-site.com
* **Reference:** MDN: CORS

### **6️ robots.txt Exposes /missions/ Directory**

* **Description:** The robots.txt file contains the /missions/ directory, which may expose **sensitive files**.
* **Impact:** Attackers can scan this directory for **hidden pages** or **admin panels**.
* **Fix:**
  + Remove sensitive directories from robots.txt.
  + Use **proper authentication** instead of hiding URLs.
* **Reference:** PortSwigger: robots.txt Exposure

### **7️ Hostname Mismatch in SSL Certificate**

* **Description:** The **SSL certificate hostname** does not match www.hackthissite.org. Instead, it points to a .onion domain.
* **Impact:** This may indicate **misconfigured SSL certificates** or **man-in-the-middle (MITM) risks**.
* **Fix:**
  + Ensure the SSL certificate **correctly matches** the domain.
  + Use **Let’s Encrypt** or a trusted CA for proper certificates.
* **Reference:** CWE-297: Improper Certificate Validation

## **3. Risk Assessment Summary**

|  |  |  |
| --- | --- | --- |
| **Vulnerability** | **Risk Level** | **Fix Required?** |
| No X-Frame-Options Header | **High** | ✅ Yes |
| BREACH Attack Possible | **High** | ✅ Yes |
| Cookies Missing Secure & HttpOnly | **Medium** | ✅ Yes |
| No X-Content-Type-Options | **Medium** | ✅ Yes |
| CORS (Access-Control-Allow-Origin: \*) | **Low** | ⚠️ Recommended |
| robots.txt Exposes /missions/ | **Low** | ⚠️ Recommended |
| SSL Hostname Mismatch | **Low** | ⚠️ Recommended |

## **4. Conclusion & Next Steps**

The scan identified **critical security issues** that should be addressed immediately. The **most urgent** fixes include:

* **Adding security headers** (X-Frame-Options, X-Content-Type-Options)
* **Fixing insecure cookies**
* **Disabling HTTP compression (BREACH attack prevention)**