

Development of an AI-Integrated Automated Scanning Tool for Security Analysis and Remediation

Name : Jackie Kim

Student ID : 300376300

Course : CSIS4495-001 Applied Research Project

Video link : [JackieK_Midterm_Video.mp4](#)

<Introduction>

In today's digital landscape, cybersecurity threats are evolving at an unprecedented pace, requiring organizations to implement robust security measures to safeguard sensitive data. One of the most effective approaches to assessing and improving cybersecurity defenses is penetration testing.

Penetration testing is an authorized, simulated cyberattack on a computer system designed to evaluate its security posture. By employing various tools, techniques, and methodologies, penetration testers mimic the actions of real attackers to uncover vulnerabilities that could be exploited (Black Duck Software, n.d.; Imperva, n.d.; IBM, n.d.). This process plays a crucial role in identifying security gaps, assessing risks, and enabling organizations to enhance their defensive strategies.

As businesses increasingly rely on digital platforms to store, process, and manage data, the need for proactive security measures has become paramount. Cyberattacks such as ransomware, phishing, and data breaches continue to pose significant threats to organizations across industries. Traditional security measures, such as firewalls and antivirus programs, are not always sufficient to protect against sophisticated cyber threats. Penetration testing provides a proactive approach to identifying and mitigating security risks before they can be exploited by malicious actors.

Penetration tests, often referred to as "pen tests," are categorized into three primary types. White Box Penetration Testing involves testers having full access to the system, including network maps and credentials, allowing for a comprehensive evaluation of security vulnerabilities. Black Box Penetration Testing is where testers have no prior knowledge of the system, mimicking external attackers attempting to exploit unknown weaknesses. Grey Box Penetration Testing occurs when testers receive limited information, such as login credentials, to simulate insider threats or external breaches. By identifying weaknesses in networks, applications, and systems, penetration testing supports compliance with regulatory standards such as PCI DSS, HIPAA, and GDPR. It provides qualitative and quantitative insights that inform security policies, resource allocation, and risk mitigation strategies (Black Duck Software, n.d.; Imperva, n.d.; IBM, n.d.).

Despite the effectiveness of penetration testing, organizations face several challenges in conducting these assessments. Resource constraints make traditional penetration testing time-consuming and financially demanding, making it difficult for small and medium-sized enterprises (SMEs) to implement regular testing. The dynamic nature of cyber threats means that vulnerabilities identified during a penetration test can quickly become outdated. Organizations may struggle to analyze and prioritize remediation efforts due to the complexity of penetration testing reports. While automated tools can streamline penetration testing, they may produce false positives or fail to detect advanced threats effectively. Without a systematic and automated approach to penetration testing, organizations risk exposing their sensitive data to cyber threats, resulting in financial losses, reputational damage, and legal consequences.

This research aims to address the limitations of traditional penetration testing by exploring the development of automated internal penetration testing tools. While existing studies highlight the benefits of penetration testing, there is limited research on automating and integrating penetration testing into an organization's continuous security assessment framework. Key knowledge gaps include the effectiveness of automated penetration testing tools compared to manual testing, the impact of automation on reducing false positives and enhancing vulnerability detection, and the feasibility of integrating automated penetration testing into cybersecurity frameworks for real-time security monitoring.

This study hypothesizes that automated penetration testing can enhance cybersecurity by providing organizations with a cost-effective, scalable, and efficient approach to vulnerability assessment. By leveraging automation, businesses can conduct regular security assessments without extensive resource allocation, improve the accuracy and consistency of penetration test results, identify and remediate vulnerabilities in real-time, reducing exposure to cyber threats, and ensure compliance with industry regulations and standards. Ultimately, this research seeks to bridge the gap between traditional penetration testing and modern security needs by proposing a framework for automated

penetration testing solutions.

<Summary of Initially Proposed Research Project>

This research focuses on the Development of an AI-Integrated Automated Scanning Tool for Security Analysis and Remediation. The study follows a structured, multi-phase approach consisting of Development, Validation, and Evaluation. Each phase plays a crucial role in ensuring that the tool is practical, efficient, and reliable for security assessments in organizational environments.

The primary objective of this research is to develop and implement an automated penetration testing tool that organizations can integrate into their annual security assessments. The tool aims to provide a standardized, scalable, and efficient solution to help organizations conduct comprehensive internal penetration tests on a regular basis.

Another key objective is to validate the tool's effectiveness in identifying vulnerabilities within simulated environments. This involves assessing its accuracy, execution speed, and adaptability. By evaluating these metrics, the research ensures that the tool meets real-world security demands and adapts to evolving IT infrastructures.

The research methodology consists of five key phases: Step-by-Step Procedure Development, Tool Development, AI Integration for Remediation, Validation, and Evaluation.

In the Step-by-Step Procedure Development Phase, the essential components of penetration testing, including reconnaissance, scanning, exploitation, and reporting, are identified and structured. This phase ensures that the testing process aligns with legal, ethical, and compliance standards.

The Tool Development Phase focuses on building the penetration testing tool using programming languages such as Python or C#. The tool integrates widely used open-source security utilities, including Nmap, Nikto and Semgrep, to automate various penetration testing tasks, thereby reducing manual effort and enhancing efficiency.

The AI Integration for Remediation Phase introduces artificial intelligence to analyze penetration testing reports. The AI system categorizes vulnerabilities based on severity and provides tailored remediation strategies, minimizing the need for human interpretation. By leveraging machine learning algorithms, the AI-driven remediation system enhances the security assessment process by offering precise and actionable recommendations.

The Validation Phase rigorously tests the tool in a simulated IT environment to assess its accuracy, efficiency, and adaptability. The tool is subjected to a variety of network configurations and threat scenarios to determine its effectiveness in detecting security vulnerabilities.

The Evaluation Phase assesses the tool's performance in comparison to manual penetration testing methods. By analyzing effectiveness, usability, and reliability, this phase determines whether the automated tool can serve as a viable alternative to traditional penetration testing approaches.

The anticipated outcome of this research is an efficient, accurate, and scalable penetration testing tool that enhances an organization's cybersecurity posture. By automating reconnaissance, scanning, and exploitation tasks, the tool is expected to significantly reduce the time and effort required for penetration testing compared to manual approaches.

Furthermore, the integration of AI in the remediation phase is expected to provide intelligent, actionable recommendations for addressing vulnerabilities. The AI-driven approach will prioritize risks based on severity, suggest targeted mitigation strategies, and improve organizations' ability to respond to security threats proactively.

Overall, the research aims to develop a streamlined, intelligent penetration testing solution that enables organizations to conduct regular security assessments efficiently while enhancing their ability

to detect and mitigate cybersecurity threats.

<Changes to the Proposal>

The primary change to the project is in the timeline. Originally, the plan was to complete the automation script and begin integrating AI tools for report analysis starting from February 10. However, this phase has not yet commenced due to the extended time required for researching and identifying the proper commands for the automated scanning tool. The complexity of selecting the most effective commands and ensuring their compatibility with the tool caused delays in the development process.

Instead of progressing directly to AI integration, efforts were redirected toward setting up a real-world testing environment. A significant milestone was achieved when contact was made with the owner of a company that receives online orders and delivers food. Permission was granted to perform vulnerability scans and penetration testing on the company's website. Additionally, access to the website's source code was provided, allowing for a more in-depth security analysis. This new development expanded the scope of the automated scanning tool by incorporating code vulnerability assessments alongside traditional security testing.

As a result of these changes, the AI research and integration phase will now begin during the week of February 24 to March 1. The AI tool will be integrated to analyze the results of the automated scanning tool and generate remediation recommendations. This integration will enhance the overall functionality of the penetration testing tool by providing intelligent, context-aware mitigation strategies. Despite the delay, these adjustments are expected to improve the project's overall effectiveness by incorporating real-world application testing and expanding the range of vulnerabilities assessed.

<Project Planning and Timeline>

Starting from February 24, the project will move into Phase 3, where the investigation and testing of AI tools for remediation will take place, followed by integration with the existing penetration testing tool for automated report analysis. This phase will run until March 22, with deliverables including research findings, feasibility test reports, and the integrated AI tool. Phase 4 will begin on March 9, focusing on setting up the testing environment and evaluating the performance of the AI-integrated tool, with testing and validation concluding by March 29. The final phase, Phase 5, will run from March 23 to April 12, comparing the AI tool with manual testing, gathering qualitative feedback, and finalizing the comprehensive evaluation and project report, ensuring that the tool meets all expected goals and identifying areas for future improvement.

Task	Feb 24 - Mar 1	Mar 2 - Mar 8	Mar 9 - Mar 15	Mar 16 - Mar 22	Mar 23 - Mar 29	Mar 30 - Apr 5	Apr 6 - Apr 12
Phase 3: AI Integration for Remediation							
Research AI Tools for Remediation							
Test Feasibility and Functionality of AI Tools							
Integrate AI tools and Develop AI model for report analysis							
Test AI integration and performance							
Phase 4: Validation							
Set up testing environment							
Test and evaluate performance							
Phase 5: Evaluation							
Compare with manual testing							
Gather qualitative feedback							
Finalize evaluation and reporting							

<Implemented Feature>

This Python script automates the process of conducting penetration testing and security scanning on a specified target. After performing the security tests, it generates a detailed HTML report that summarizes the results and serves it through a simple HTTP server for easy access. The script leverages tools like Nmap, Nikto, Semgrep, and other security scanning utilities to test the target's security posture.

The `run_command` function is responsible for executing a shell command and capturing its output while also tracking the runtime. It accepts a command as a string and an optional timeout value (default is 900 seconds). If the command execution exceeds the timeout period, or if there is an error, the function handles the exceptions appropriately. Additionally, it tracks and prints the time taken to execute the command, including start and end times, duration, and the output.

```
def run_command(command, timeout=900):
    start_time = time.time()

    try:
        result = subprocess.run(command, shell=True, check=True, text=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, timeout=timeout)
        output = result.stdout
    except subprocess.TimeoutExpired:
        output = f"Timeout: Command exceeded {timeout} seconds: {command}"
    except subprocess.CalledProcessError as e:
        output = f"Error executing command: {e}"

    end_time = time.time()
    duration = end_time - start_time

    print(f"\n[✓] Command: {command}")
    print(f"    - Start: {time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(start_time))}")
    print(f"    - End: {time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(end_time))}")
    print(f"    - Duration: {duration:.2f} seconds\n")

    return output
```

The `run_semgrep` function performs a Semgrep security scan on the current working directory. Semgrep is a static analysis tool used to find vulnerabilities in code. The function constructs and runs

the Semgrep command, saving the results in a JSON file. It then loads and parses this JSON file to extract the findings. The output is returned in a structured dictionary, which is later used for generating the report.

```
def run_semgrep():
    print("\nRunning Semgrep Security Scan...\n")
    semgrep_command = f"semgrep --config=auto --json > {SEMGRP_JSON_FILE}"
    run_command(semgrep_command)

    try:
        with open(SEMGRP_JSON_FILE, "r") as file:
            data = json.load(file)
    except FileNotFoundError:
        print("Error: Semgrep report file not found.")
        return {}

    sorted_results = {}
    for result in data.get("results", []):
        if "comment" in result["extra"].get("metadata", {}).get("category", "").lower():
            continue

        category = result["check_id"].split(".")[1]
        file_path = result["path"]
        line_number = result["start"]["line"]

        if category not in sorted_results:
            sorted_results[category] = {"severity": result["extra"].get("severity", "Unknown"), "files": {}}

        if file_path not in sorted_results[category]["files"]:
            sorted_results[category]["files"][file_path] = []

        sorted_results[category]["files"][file_path].append(line_number)

    return sorted_results
```

The `generate_html_report` function creates a comprehensive HTML report that consolidates the results of the penetration tests and Semgrep security scan. The report is structured using an HTML template that includes basic styles and a script to toggle the visibility of long command outputs. The Semgrep findings are included in a dedicated section at the end of the report, where vulnerabilities are grouped by category and severity. This function ultimately writes the generated HTML content to a file, making it accessible for review. The generated report is designed to be interactive and user-friendly.

```
def generate_html_report(results, semgrep_results):
    html_template = """...

    sections = ""
    for title, output in results.items():
        sections += """
        <div class="section">
            <h2>{}</h2>
            <pre>{}</pre>
        </div>
        """.format(
            title,
            output
        )

    sections += "<h2>Semgrep Security Findings</h2>"
    for category, data in semgrep_results.items():
        sections += "<h3>{} (Severity: {})</h3>".format(category, data["severity"])
        for file, lines in data["files"].items():
            sections += "<p><b>{}</b>: Lines {}</p>".format(file, '.join(map(str, lines)))

    html_template += sections + """
    </div>
</body>
</html>
"""

    with open(REPORT_FILE, "w") as f:
        f.write(html_template)
```

The main function is the central control flow of the script, orchestrating the execution of various security tests and the generation of the final report. First, it prompts the user to input the IP address or

domain of the target to scan. A series of predefined Nmap and other security scan commands are executed using the `run_command` function, which gathers information about the target, such as open ports, operating system details, and vulnerabilities. After completing these scans, the function invokes `run_semgrep` to perform a static analysis of the codebase in the working directory. The results from both the penetration testing and Semgrep scan are passed to the `generate_html_report` function to create the report.

```
def main():
    print("\nStarting automated penetration testing...\n")
    target = input("Enter the IP address or domain to scan: ")

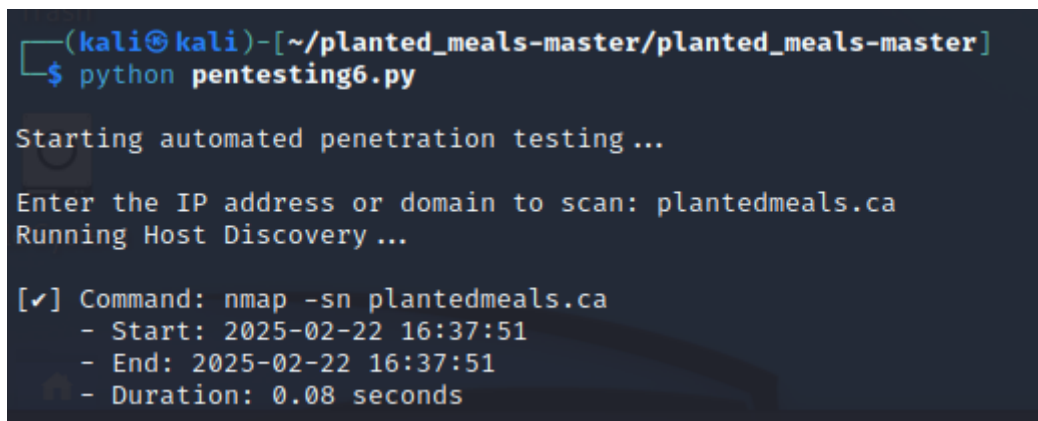
    commands = {
        "Host Discovery": f"nmap -sn {target}",
        "Full Port Scan": f"nmap -p- -T4 {target}",
        "OS and Version Detection": f"sudo nmap -O -A {target}",
        "Web Enumeration": f"nmap --script=http-title,http-headers,http-methods -p 80,443 {target}",
        "DNS Enumeration": f"nmap --script=dns-brute,dns-zone-transfer -p 53 {target}",
        "Vulnerability Scan": f"nmap --script=vulners -sV {target}",
        "Firewall Evasion Scan": f"nmap -f --mtu 16 {target}",
        "Nikto Scan (Basic)": f"nikto -h {target}",
        "Nikto Scan (SSL)": f"nikto -h {target} -ssl",
        "WhatWeb Scan": f"whatweb {target}",
    }

    results = {}
    for title, cmd in commands.items():
        print(f"Running {title}...")
        results[title] = run_command(cmd)

    semgrep_results = run_semgrep()

    generate_html_report(results, semgrep_results)
    print(f"\nReport generated. The report is saved in {REPORT_FILE}")
```

Following is the output when you run the Python tool for the automated penetration testing scanning tool. Upon execution, the program prompts the user to input the IP address or domain to scan. Once the user enters the domain name, as shown in the screenshot, the tool begins running the predefined scanning commands one by one. For each command, the tool displays the command being executed along with its start time, end time, and the total duration of the execution. This process ensures that the user is informed about the progress and timing of each scan throughout the testing procedure.



```
(kali㉿kali)-[~/planted_meals-master/planted_meals-master]
$ python pentesting6.py

Starting automated penetration testing...

Enter the IP address or domain to scan: plantedmeals.ca
Running Host Discovery ...

[✓] Command: nmap -sn plantedmeals.ca
    - Start: 2025-02-22 16:37:51
    - End: 2025-02-22 16:37:51
    - Duration: 0.08 seconds
```

At the end of the run, the program provides a file name of the generated report. This report contains the comprehensive scanning results, including details of the performed penetration tests and Semgrep security findings.

```
Running Semgrep Security Scan ...

[✓] Command: semgrep --config=auto --json > semgrep_report.json
  - Start: 2025-02-23 17:30:21
  - End: 2025-02-23 17:31:46
  - Duration: 85.58 seconds

Report generated. The report is saved in report.html
```

Following is a screenshot of the penetration test report for the scan conducted on plantedmeals.ca. The report provides insights into the host's discoverable services, potential vulnerabilities, and associated network details. For a more detailed analysis and full findings, you can refer to the complete report available as a report.html file on the GitHub repository.

The Host Discovery scan shows that the target, plantedmeals.ca with IP address 199.34.228.73, is active and reachable. The scan results show that the host responded with a low latency of 0.010 seconds. Additionally, a reverse DNS lookup reveals that the host is associated with the domain pages-custom-25.weebly.com.

Host Discovery

```
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-23 17:13 PST
Nmap scan report for plantedmeals.ca (199.34.228.73)
Host is up (0.010s latency).
rDNS record for 199.34.228.73: pages-custom-25.weebly.com
Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

The Full Port Scan identifies several open ports on the target host. Notably, ports 80 (HTTP) and 443 (HTTPS) are open, which are standard for web traffic. The scan also lists several other open ports, such as 2052 (clearvisn), 2053 (knetd), and 8080 (HTTP proxy), indicating that the target host may be running multiple services. The scan also shows that a large number of ports (65522) are filtered, meaning they did not respond to the scan, likely due to a firewall or other security measures.

Full Port Scan

```
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-23 17:13 PST
Nmap scan report for plantedmeals.ca (199.34.228.73)
Host is up (0.017s latency).
rDNS record for 199.34.228.73: pages-custom-25.weebly.com
Not shown: 65522 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
2052/tcp  open  clearvisn
2053/tcp  open  knetd
2082/tcp  open  infowave
2083/tcp  open  radsec
2086/tcp  open  gnunet
2087/tcp  open  eli
2095/tcp  open  nbx-ser
2096/tcp  open  nbx-dir
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
8880/tcp  open  cddbp-alt

Nmap done: 1 IP address (1 host up) scanned in 98.04 seconds
```


OS and Version Detection

Web Enumeration

```
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-23 17:16 PST
Nmap scan report for plantedmeals.ca (199.34.228.73)
Host is up (0.0086s latency).
rDNS record for 199.34.228.73: pages-custom-25.weebly.com

PORT      STATE SERVICE
80/tcp    open  http
|_ http-headers:
|   Date: Mon, 24 Feb 2025 01:16:59 GMT
|   Content-Type: text/html; charset=iso-8859-1
|   Transfer-Encoding: chunked
|   Connection: close
|   Location: http://www.plantedmeals.ca/
|   CF-Ray: 916b9d07c880491d-VYR
|   CF-Cache-Status: BYPASS
|   Vary: Accept-Encoding
|   Set-Cookie: __cf_bm=10AhmwsObL.okKs5VbfFSnouYX1ZCxoq.xMo0JAKAag-1740359819-1.0.1.1-FYXEPQR6hdCxaED0VY019
|   Server: cloudflare
|_ (Request type: GET)
|_ http-title: Did not follow redirect to http://www.plantedmeals.ca/
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
443/tcp    open  https
|_ http-title: Did not follow redirect to https://www.plantedmeals.ca/
|_ http-methods:
|_   Supported Methods: GET HEAD POST OPTIONS
|_ http-headers:
|   Date: Mon, 24 Feb 2025 01:16:59 GMT
|   Content-Type: text/html; charset=iso-8859-1
|   Transfer-Encoding: chunked
|   Connection: close
|   Location: https://www.plantedmeals.ca/
|   CF-Ray: 916b9d06eb8b2da7-VYR
|   CF-Cache-Status: BYPASS
|   Vary: Accept-Encoding
|   Set-Cookie: __cf_bm=WYf.0v_LGVnG.Q1LgUwqgAMQkYnagmFsUymK9wNYNcM-1740359819-1.0.1.1-hFPpoEgDqtaAM5KxRPaBF
|   Server: cloudflare
|_ (Request type: GET)

Nmap done: 1 IP address (1 host up) scanned in 1.21 seconds
```

The DNS Enumeration scan shows that the DNS service on port 53 is filtered, meaning it did not respond to the scan. However, the scan successfully identified several subdomains of plantedmeals.ca using DNS brute-forcing. These subdomains include admin.plantedmeals.ca and shop.plantedmeals.ca, along with their associated IP addresses. This information could potentially be useful for discovering additional services or administrative interfaces associated with the target.

DNS Enumeration

```
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-23 17:16 PST
Nmap scan report for plantedmeals.ca (199.34.228.73)
Host is up (0.019s latency).
rDNS record for 199.34.228.73: pages-custom-25.weebly.com

PORT      STATE SERVICE
53/tcp    filtered domain

Host script results:
|_ dns-brute:
|   DNS Brute-force hostnames:
|   admin.plantedmeals.ca - 13.248.131.213
|   admin.plantedmeals.ca - 15.197.152.254
|   admin.plantedmeals.ca - 3.33.161.45
|   admin.plantedmeals.ca - 35.71.150.51
|   www.plantedmeals.ca - 199.34.228.73
|   shop.plantedmeals.ca - 13.248.131.213
|   shop.plantedmeals.ca - 15.197.152.254
|   shop.plantedmeals.ca - 3.33.161.45
|_   shop.plantedmeals.ca - 35.71.150.51

Nmap done: 1 IP address (1 host up) scanned in 1.52 seconds
```

The Vulnerability Scan provides information about the open ports and the services running on the target. It shows that HTTP (port 80), HTTPS (port 443), and other ports like 8080 and 8443 are open, all running Cloudflare's HTTP proxy. The scan does not specifically highlight vulnerabilities but does provide details on the services detected, which may require further analysis for security testing or patching.

Vulnerability Scan

```
Starting Nmap 7.92 ( https://nmap.org ) at 2025-02-23 17:17 PST
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 41.40% done; ETC: 17:17 (0:00:03 remaining)
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 43.00% done; ETC: 17:17 (0:00:03 remaining)
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 43.85% done; ETC: 17:17 (0:00:03 remaining)
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 78.10% done; ETC: 17:17 (0:00:01 remaining)
Nmap scan report for plantedmeals.ca (199.34.228.73)
Host is up (0.014s latency).
rDNS record for 199.34.228.73: pages-custom-25.weebly.com
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
80/tcp    open  http      Cloudflare http proxy
|_http-server-header: cloudflare
443/tcp   open  ssl/http  Cloudflare http proxy
|_http-server-header: cloudflare
8080/tcp  open  http      Cloudflare http proxy
|_http-server-header: cloudflare
8443/tcp  open  ssl/http  Cloudflare http proxy
|_http-server-header: cloudflare

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.39 seconds
```

The basic Nikto scan was run on Target IP: 199.34.228.73, with the hostname plantedmeals.ca at port 80, which is the standard HTTP port. The website's server is hosted by Cloudflare. The scan detected that certain security headers were missing, such as the anti-clickjacking X-Frame-Options header and the X-XSS-Protection header. Both headers are important for preventing certain types of attacks, such as clickjacking and cross-site scripting (XSS). The X-Content-Type-Options header was also missing, which could allow browsers to incorrectly render content based on MIME types.

The scan also found that the website's root page (/) was redirected to the URL <http://www.plantedmeals.ca/>. Additionally, some cookies, such as language and cookie-consent, were created without the HttpOnly flag, potentially exposing them to cross-site scripting (XSS) attacks. Other headers were found, including surrogate-control and x-host, which are not commonly seen in standard server configurations and could have security implications.

Nikto Scan (Basic)

- Nikto v2.1.6

```
+ Target IP:      199.34.228.73
+ Target Hostname: plantedmeals.ca
+ Target Port:    80
+ Start Time:     2025-02-23 17:17:32 (GMT-8)

+ Server: cloudflare
+ IP address found in the '__cf_bm' cookie. The IP is "1.0.1.1".
+ IP address found in the 'set-cookie' header. The IP is "1.0.1.1".
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the
+ Root page / redirects to: http://www.plantedmeals.ca/
+ IP address found in the 'report-to' header. The IP is "1.0.1.1".
+ IP address found in the 'content-security-policy-report-only' header. The IP is "1.0.1.1".
+ Uncommon header 'report-to' found, with contents: {"endpoints":[{"url":"https://csp-reporting.cloudflare
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Uncommon header 'surrogate-control' found, with contents: max-age=60
+ Cookie language created without the httponly flag
+ Cookie cookie-consent created without the httponly flag
+ Uncommon header 'x-host' found, with contents: blu135.sf2p.intern.weebly.net
+ 7785 requests: 0 error(s) and 12 item(s) reported on remote host
+ End Time:       2025-02-23 17:29:13 (GMT-8) (701 seconds)

+ 1 host(s) tested
```

The website was found to use an SSL certificate issued by Let's Encrypt (CN=R10), and the encryption protocol in use was TLS_AES_256_GCM_SHA384. The server is again hosted by Cloudflare. The scan revealed the absence of a Strict-Transport-Security (HSTS) header. This header is important for enforcing secure HTTPS connections and preventing downgrade attacks. Similarly, the Expect-CT header was not set. This header is useful for preventing the use of mis-issued SSL certificates. Like the basic HTTP scan, the X-Content-Type-Options header was also missing, leaving the site vulnerable to certain types of attacks that exploit content type misinterpretation.

Nikto Scan (SSL)

- Nikto v2.1.6

```
+ Target IP:      199.34.228.73
+ Target Hostname: plantedmeals.ca
+ Target Port:    443

+ SSL Info:       Subject: /CN=www.plantedmeals.ca
                  Ciphers: TLS_AES_256_GCM_SHA384
                  Issuer:  /C=US/O=Let's Encrypt/CN=R10
+ Start Time:     2025-02-23 17:29:14 (GMT-8)

+ Server: cloudflare
+ IP address found in the '__cf_bm' cookie. The IP is "1.0.1.1".
+ IP address found in the 'set-cookie' header. The IP is "1.0.1.1".
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
+ The site uses SSL and Expect-CT header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the
+ Root page / redirects to: https://www.plantedmeals.ca/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Uncommon header 'surrogate-control' found, with contents: max-age=60
+ Scan terminated: 20 error(s) and 8 item(s) reported on remote host
+ End Time:       2025-02-23 17:30:11 (GMT-8) (57 seconds)

+ 1 host(s) tested
```

The WhatWeb scan revealed that the website is located at <http://plantedmeals.ca>, which redirects to <http://www.plantedmeals.ca/>. The presence of this redirect could potentially confuse search engines

and visitors. The scan identified Cloudflare as the server hosting the website and detected Weebly as the platform behind the site.

A Google Analytics tracking ID (UA-7870337-1) was found, suggesting that the website tracks user interactions. The cookies associated with the site, such as `__cf_bm`, were also noted, which are commonly used by Cloudflare for bot management. The website responded with a 301 Moved Permanently status, indicating that it is configured to redirect visitors to another URL.

Other uncommon headers identified by the scan included `cf-ray` and `cf-cache-status`, which are related to Cloudflare's content delivery and caching system. The page's title was reported as "Planted Meals - Plant-Based Meal Delivery Service & Greater Vancouver Area", reflecting the business focus of the website.

WhatWeb Scan

```
+[1m+[34mhttp://plantedmeals.ca/[0m [301 Moved Permanently] +[1mCookies+[0m+[0m+[22m__cf_bm+[0m], +[1mCount
+[1m+[34mhttp://www.plantedmeals.ca/[0m [301 Moved Permanently] +[1mCookies+[0m+[0m+[22m__cf_bm,is_mobile+
+[1m+[34mhttps://www.plantedmeals.ca/[0m [200 OK] +[1mCloudFlare+[0m, +[1mCookies+[0m+[0m+[22m__cf_bm,is_m
```

```
(kali@kali)-[~/planted_meals-master/planted_meals-master]
$ whatweb plantedmeals.ca
http://plantedmeals.ca [301 Moved Permanently] Cookies[__cf_bm], Country[UNITED STATES][US], HTTPServer[cloudflare], HttpOnly[__cf_bm], IP[199.34.228.73], RedirectLocation[http://www.plantedmeals.ca/], Title[301 Moved Permanently], UncommonHeaders[cf-ray,cf-cache-status]
http://www.plantedmeals.ca/ [301 Moved Permanently] Cookies[__cf_bm,is_mobile], Country[UNITED STATES][US], HTML5, HTTPServer[cloudflare], HttpOnly[__cf_bm], IP[199.34.228.73], Meta-Refresh-Redirect[https://www.plantedmeals.ca/], RedirectLocation[https://www.plantedmeals.ca/], Title[Redirecting to https://www.plantedmeals.ca/], UncommonHeaders[cf-ray,cf-cache-status,x-host], Weebly, X-Host[grn111.sf2p.intern.weebly.net], X-UA-Compatible[IE=edge,chrome=1]
https://www.plantedmeals.ca/ [200 OK] CloudFlare, Cookies[__cf_bm,is_mobile,language], Country[UNITED STATES][US], Google-Analytics[UA-7870337-1], HTML5, HTTPServer[cloudflare], HttpOnly[__cf_bm], IP[199.34.228.73], JQuery[1.8.3], Open-Graph-Protocol, Script[text/javascript], Title[Planted Meals - Plant-Based Meal Delivery Service & Greater Vancouver Area], UncommonHeaders[cf-ray,cf-cache-status,x-host], Weebly, X-Host[grn91.sf2p.intern.weebly.net], X-UA-Compatible[IE=edge,chrome=1]
```

The Semgrep security analysis identified potential vulnerabilities in the website's Ruby on Rails controllers and views. Issues such as improper input handling, query sanitization flaws, and session management concerns were flagged, raising the risk of SQL injection, session hijacking, and unauthorized access. Vulnerabilities were particularly noted in areas like gift cards, allergies, and orders. While the severity of these findings was categorized as a warning, they should still be addressed to strengthen the site's security.

Semgrep Security Findings

rails (Severity: WARNING)

`app/controllers/admin_bag_trackers_controller.rb`: Lines 10, 14

`app/controllers/admin_gift_cards_controller.rb`: Lines 24, 28, 38, 43

`app/controllers/admin_orders_controller.rb`: Lines 15, 32, 33, 37, 41, 48, 54

`app/controllers/admin_referrals_controller.rb`: Lines 22, 35, 39

`app/controllers/allergies_controller.rb`: Lines 18, 34, 36

The scan also detected a few language-related issues and browser behavior warnings. Specifically, files like `bag_trackers_controller.rb` and `orders_controller.rb` were noted for possible problems with localization, which could lead to a lack of language-specific handling or a failure to adapt to different regional formats.

On the browser side, there were issues detected with JavaScript file processing. It appears that some scripts might not be optimized, leading to potential performance issues or conflicts in how the website's functionality is rendered. Caching mechanisms were also flagged, suggesting that there may

be opportunities for optimization in how static resources (like scripts and styles) are managed.

lang (Severity: WARNING)

app/controllers/bag_trackers_controller.rb: Lines 6

app/controllers/orders_controller.rb: Lines 102

coverage/assets/0.12.2/application.js: Lines 1, 1, 1, 3, 4, 4, 4, 5, 5

coverage/assets/0.12.3/application.js: Lines 1, 1, 1, 3, 4, 4, 4, 5, 5

pentesting6.py: Lines 18

browser (Severity: ERROR)

coverage/assets/0.12.2/application.js: Lines 3, 4

coverage/assets/0.12.3/application.js: Lines 3, 4

<Work Date/Hours logs>

Date	Number of Hours	Description of Work Done
10-Feb-25	2	Gathered tools and commands which I can use for automated scanning tool (Nmap, OpenVas, Metasploit Framework)
12-Feb-25	2	Gathered tools and commands which I can use for automated scanning tool (Nikto, Xprobe2, SMBClient)
13-Feb-25	2	Gathered tools and commands which I can use for automated scanning tool (Whatweb, Hydra, SonarQube)
14-Feb-25	1.5	Tested each commands in Linux and identified which components or information can be generated as outcome (only Nmap)
15-Feb-25	4	Tested each commands in Linux and identified which components or information can be generated as outcome (rest of the tools)
18-Feb-25	2	python file to execute commands was created
19-Feb-25	4	Python file was updated to include more commands, Metasploit framework tool was tested
20-Feb-25	4	Python file was updated and decided to eliminate p0f tool since it is redundant and it is constantly causing error, SonarQube tool was tested for code check. However, I failed to use this tool and couldn't make it work. I switched to use another tool called "Semgrep".
22-Feb-25	4	Debugged the code for automated scanning tool and started generating reports.
23-Feb-25	6	Final check with the tool and midterm report generation

<References>

1. Black Duck Software. (n.d.). What is penetration testing? Black Duck Software. Retrieved January 23, 2025, from <https://www.blackduck.com/glossary/what-is-penetration-testing.html>
2. Imperva. (n.d.). Penetration testing. Imperva. Retrieved January 23, 2025, from <https://www.imperva.com/learn/application-security/penetration-testing/>
3. IBM. (n.d.). Penetration testing: What it is and why it's important. IBM. Retrieved January 23, 2025, from <https://www.ibm.com/think/topics/penetration-testing>