

Evaluating Web Application and Server Security: A Hands-On Test with Wapiti and Nikto

Aidan Ingram

University of Kansas, Department of Electrical Engineering and Computer Science

Abstract

The purpose of this report is to explore the functionality of two vulnerability scanner tools, Nikto and Wapiti, which have not been covered in EECS 465. This report focuses on web application and server vulnerabilities, with a detailed analysis of the tools' inner workings, technical mechanisms, and effectiveness in identifying vulnerabilities. Through hands-on testing, this report also aims to provide insights into Nikto and Wapiti's capabilities and limitations in detecting web server and application vulnerabilities.

Contents

1	Introduction	1
1.1	Web Security - Importance . . .	1
2	Motivation	2
3	Tools and Methodology	2
3.1	Nikto Web Scanner	2
3.2	Wapiti Web Scanner	3
3.3	Methodology	3
4	Testing and Evaluation	4
4.1	Nikto Testing Process	4
4.2	Wapiti Testing Process	4
4.3	Test Environment and Documentation	5
5	Hands-On Demo (Figures/Data)	5
5.1	Nikto Testing Results	5
5.2	Verification of Findings	6
5.3	Wapiti Testing Results	7
5.4	Verification of Findings	8
6	Results and Discussion	9
7	Conclusion	9

1 Introduction

Web applications are prime targets for attackers due to their exposure on the internet and

the sensitive data they handle. This report explores two vulnerability scanner tools, **Nikto** and **Wapiti**, used to detect vulnerabilities in web applications and servers. These tools help identify issues like outdated software, cross-site scripting (XSS), and SQL injection, which are critical to address for maintaining web security.

My report evaluates how these tools perform in practice, and compare their usage. Using isolated virtual machine setups with **Metasploitable** and **Damn Vulnerable Web Application (DVWA)**, I will assess the tools' effectiveness in detecting vulnerabilities.

1.1 Web Security - Importance

An attack on a vulnerability as simple as SQL injection could cause the leakage of very sensitive data, directly affecting users. Vulnerability scanners like Nikto and Wapiti are essential tools in identifying and mitigating these risks, enabling organizations to proactively address security flaws and strengthen their defenses. These tools help uncover weaknesses that could be exploited, reducing the potential impact of cyber threats. To further elaborate on the necessity of this report, refer to section 2, the motivation.

2 Motivation

Web server and application vulnerabilities are of significant concern to modern cybersecurity professionals, and rightfully so. With the ever-expanding nature of the internet and the vast amounts of data being added to it, there is no sign of this growth slowing down. As reported in recent studies on internet traffic volume^[6], web traffic has been steadily increasing. Between 2010 and 2020, total traffic volume grew from 21.5 Exabytes to 222.4 Exabytes per month. More notably, as of 2025, monthly internet traffic has reached 549.2 Exabytes.

This growth parallels Moore’s Law, which suggests that the number of transistors on a computer chip doubles every two years. In a similar vein, some researchers^[3] have noted that internet traffic appears to be doubling at roughly the same rate. If this trend continues, we (meaning society) may see internet traffic double every five years, posing significant challenges for maintaining a secure, scalable internet infrastructure. While Moore’s Law itself may be nearing an asymptote due to hardware limitations, the demand for increasing network capacity and security remains high.

As cybersecurity professionals, it is our responsibility to safeguard the expanding internet. With the web’s rapid growth, tools like **Nikto** and **Wapiti** are essential for minimizing security exploits and maintaining a secure internet. Figure 1 illustrates this trend in internet traffic growth.

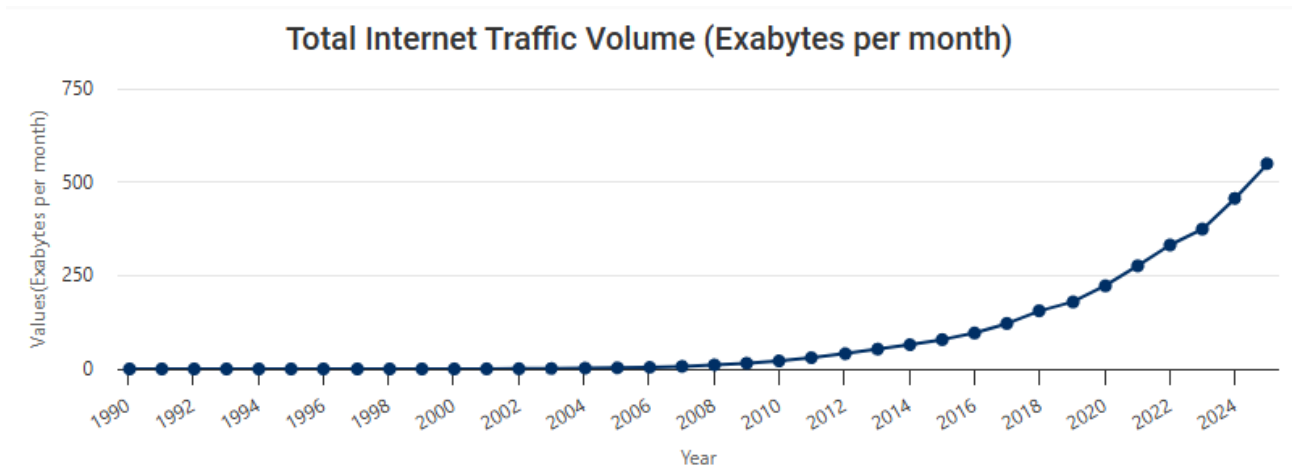


Figure 1: Graph showing internet traffic volume

3 Tools and Methodology

This section outlines the tools used for vulnerability scanning (Nikto and Wapiti) and the methodologies followed to identify and verify vulnerabilities within the target web applications.

3.1 Nikto Web Scanner

Nikto is an open-source web server scanner designed to detect various vulnerabilities such as outdated software versions, misconfigurations, and missing HTTP security headers. It can be hard to understand initially, so check out Figure 2. Nikto operates by:

- **Fingerprinting Web Technologies:** Nikto identifies the underlying web server and application technologies through HTTP response analysis, allowing it to detect known vulnerabilities in outdated software versions^[7].
- **Vulnerability Databases:** The scanner compares responses against a database of known vulnerabilities to detect issues like missing security headers (e.g., `X-Frame-Options`) and outdated software.
- **Automated Testing:** Nikto performs automated tests for common vulnerabilities like SQL injection, XSS, and directory traversal by sending crafted payloads to the server and analyzing the responses^{[5][10]}.

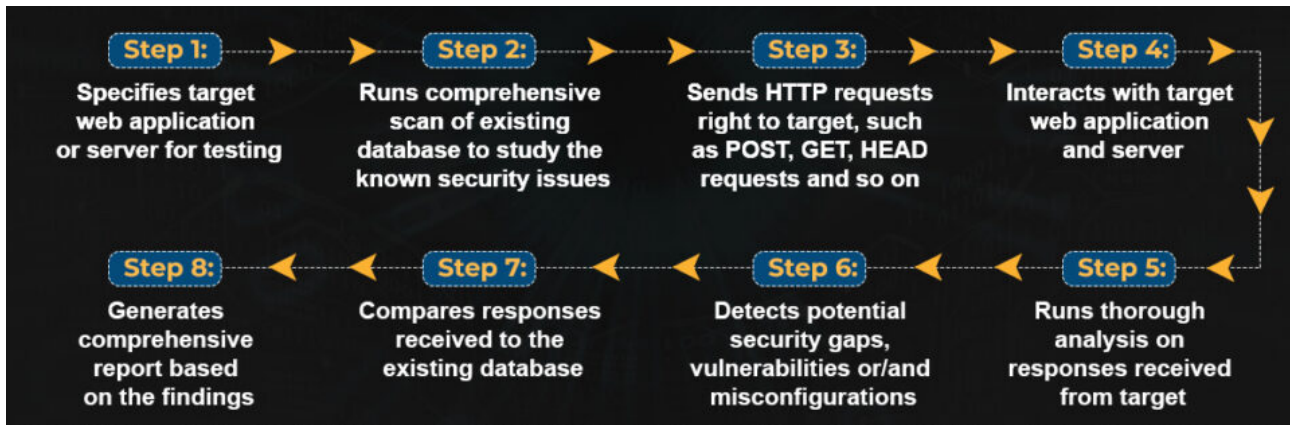


Figure 2: Overview of the Nikto scanning process.^[2]

3.2 Wapiti Web Scanner

Wapiti is a dynamic application security scanner that focuses on detecting vulnerabilities in web applications, particularly injection flaws, session management issues, and improper HTTP header configurations. While not overwhelmingly so, Wapiti is consistently more simple than Nikto (likely due to no server interaction). Key features include:

- **Crawl and Scan:** Wapiti first crawls the target application to map out its structure before performing tests for common vulnerabilities such as SQL injection and XSS^[1].
- **Injection Testing:** It uses a "blind" testing method, injecting different payloads into the application and analyzing the responses to identify vulnerabilities.
- **Cookie and Session Analysis:** Wapiti examines cookies for security issues like the absence of the `HttpOnly` flag, which could lead to session hijacking^[8], a very serious cybercrime.

3.3 Methodology

The scanning process involved using Nikto and Wapiti in parallel to identify and verify vulnerabilities within the target web applications.

- **Nikto Scan:** The web server was scanned using Nikto to detect outdated software and missing security headers. The results provided insights into potential server-side vulnerabilities, such as the absence of key HTTP headers.
- **Wapiti Scan:** Wapiti was used to assess the application for client-side vulnerabilities, including injection flaws, improper cookie handling, and missing Content Security Policies (CSP).

By using both tools, I ensured comprehensive coverage of both server-side and application-level vulnerabilities.

4 Testing and Evaluation

For this study, I selected Nikto and Wapiti to assess the vulnerabilities of web servers and applications. The testing environment consists of **Metasploitable** (a vulnerable web server) and **BWA** (a vulnerable web application), both running within an isolated virtual machine network on Kali Linux.

4.1 Nikto Testing Process

Nikto is a web server scanner designed to identify common vulnerabilities such as outdated software, insecure HTTP methods, and misconfigurations^[7].

- **Configuration:** Install Nikto on Kali using:

```
sudo apt-get install nikto
```

- **Testing:** Run a standard scan on Metasploitable's web server:

```
nikto -h http://<Metasploitable_IP>
```

- **Expected Results:** Nikto will identify common web server issues such as outdated software or insecure configurations.
- **Verification:** After scanning, I will verify the identified issues by checking the web server configuration or exploiting the vulnerabilities presented in the tool's usage.

4.2 Wapiti Testing Process

Wapiti is a web application scanner that targets vulnerabilities such as SQL Injection, XSS, and file inclusion^[1].

- **Configuration:** Install Wapiti on Kali using:

```
sudo apt-get install wapiti
```

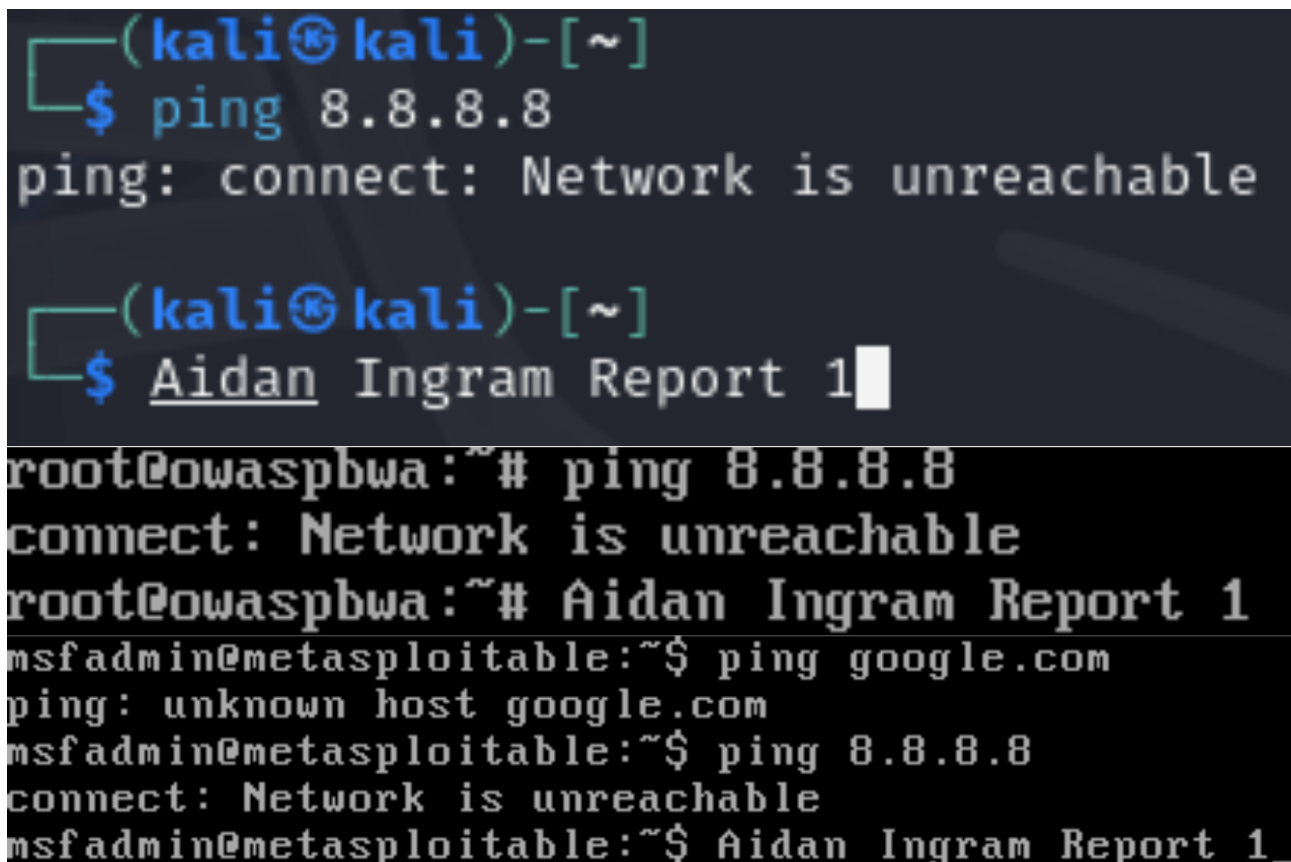
- **Testing:** Run a scan on the DVWA application:

```
wapiti -u http://<BWA_IP>/dvwa/
```

- **Expected Results:** Wapiti will identify web application vulnerabilities such as SQL Injection, XSS, and file inclusion.
- **Verification:** After scanning, I will verify the reported vulnerabilities by attempting to exploit them or checking for evidence in the application code.

4.3 Test Environment and Documentation

Both tools are tested within an isolated virtual machine setup, refer to Figure 3 for proof. Test results are documented by capturing screenshots of tool outputs, listing detected vulnerabilities, and verifying findings through manual exploitation. This allows for an assessment of the accuracy and effectiveness of both Nikto and Wapiti in identifying real-world vulnerabilities. I have done this in the next section.



```
(kaliⓂkali)-[~]
└─$ ping 8.8.8.8
ping: connect: Network is unreachable

(kaliⓂkali)-[~]
└─$ Aidan Ingram Report 1

root@owaspbwa:~# ping 8.8.8.8
connect: Network is unreachable
root@owaspbwa:~# Aidan Ingram Report 1

msfadmin@metasploitable:~$ ping google.com
ping: unknown host google.com
msfadmin@metasploitable:~$ ping 8.8.8.8
connect: Network is unreachable
msfadmin@metasploitable:~$ Aidan Ingram Report 1
```

Figure 3: Proof my methodology is restricted to my environment.

5 Hands-On Demo (Figures/Data)

This section presents figures related to the running of Nikto and Wapiti in my environment.

5.1 Nikto Testing Results

The Nikto scan results from the Metasploitable web server identified the following critical vulnerabilities, as can be seen in Figure 4:

- **Outdated Software:**

- Apache version: 2.2.8 (Outdated, current version is at least Apache 2.4.54).
- PHP version: 5.2.4-2ubuntu5.10 (Outdated, current version is at least PHP 8.1.5).

Impact: These outdated versions of Apache and PHP may contain known vulnerabilities that have been fixed in later versions, such as authentication bypass^[4].

- **Missing HTTP Security Headers:**

- X-Frame-Options header is not present (anti-clickjacking protection).
- X-Content-Type-Options header is not set (could allow MIME type confusion).

Impact: The absence of these headers can make the application vulnerable to clickjacking and content-type spoofing attacks.

```
(kali@kali) [~]
└─$ nikto -h http://192.168.56.101
- Nikto v2.5.0

+ Target IP: 192.168.56.101
+ Target Hostname: 192.168.56.101
+ Target Port: 80
+ Start Time: 2025-03-09 19:57:50 (GMT-4)

+ Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch
+ /: Server may leak inodes via ETags, header found with file /, inode: 67575, size: 45, mtime: Wed Mar 17 10:08:25 2010. See:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different
fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type
-header/
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ PHP/5.2.4-2ubuntu5.10 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following
alternatives for 'index' were found: index.html. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.x
force.ibmcloud.com/vulnerabilities/8275
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ PHP/5.2 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10.
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See:
CWE-552
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /tikiwiki/tiki-graph_formula.php?w=16h-16s-16min-16max-26f[]=x.tan.phpinfo()&t=png&title=http://blog.cirt.net/rfiinc.txt: Cookie
PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /tikiwiki/tiki-graph_formula.php: Output from the phpinfo() function was found.
+ /tikiwiki/tiki-graph_formula.php?w=16h-16s-16min-16max-26f[]=x.tan.phpinfo()&t=png&title=http://blog.cirt.net/rfiinc.txt: Tiki
wiki contains a vulnerability which allows remote attackers to execute arbitrary PHP code. See: http://cve.mitre.org/cgi-bin
/cvename.cgi?name=CVE-2007-5423
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8909 requests: 1 error(s) and 19 item(s) reported on remote host
+ End Time: 2025-03-09 19:58:21 (GMT-4) (31 seconds)

+ 1 host(s) tested
```

Figure 4: Nikto scanning output after tool use.

5.2 Verification of Findings

To verify the vulnerabilities reported by Nikto, the following actions were taken:

- **Outdated Software:**

- The Apache version was checked by running `apache2 -v`, confirming the version was 2.2.8.
- The PHP version was confirmed using `php -v`, which returned PHP 5.2.4-2ubuntu5.10.

- **Missing HTTP Security Headers:**

- The HTTP headers were examined using the following command:

```
curl -I http://192.168.56.101
```

The headers `X-Frame-Options` and `X-Content-Type-Options` were not present in the response.

5.3 Wapiti Testing Results

The Wapiti scan results from the DVWA web application identified the following critical vulnerabilities (please note Figure's 5 and 6 represent this information):

- **Content Security Policy (CSP):**

- CSP is not set on the web application.

Impact: This lack of CSP exposes the application to Cross-Site Scripting (XSS) and data injection attacks^[9].

- **HTTP Secure Headers:**

- Missing security headers: `X-Frame-Options`, `X-XSS-Protection`, `X-Content-Type-Options`, and `Strict-Transport-Security`.

Impact: The absence of these headers leaves the application vulnerable to clickjacking, XSS attacks, MIME type confusion, and man-in-the-middle attacks.

- **HttpOnly Flag Cookie:**

- The `HttpOnly` flag is not set on cookies (`PHPSESSID`, `security`).

Impact: Without the `HttpOnly` flag, client-side scripts can access cookies, potentially leading to session hijacking.



```
root@kali:~# curl -I http://192.168.56.101/dvwa/
WAPITIE
wapiti-3.0-4 (wapiti.sourceforge.io)
[*] Wapiti found 2 URLs and forms during the scan
[*] Loading modules:
    Backup, blindsql, brute_login_form, buster, cookieflags, crlf, csp, csrf, exec, file, htaccess, http_headers, methods
    , sirt0, parametxss, redirect, shellhack, sql, csrf, wmap, xss, xxe
Problem with local app database.
Downloading from the web...
[Error downloading wmap database.
[*] Launching module csp
CSP is not set
[*] Launching module http_headers
Checking X-Frame-Options :
X-Frame-Options is not set
Checking X-XSS-Protection :
X-XSS-Protection is not set
Checking X-Content-Type-Options :
X-Content-Type-Options is not set
Checking Strict-Transport-Security :
Strict-Transport-Security is not set
[*] Launching module cookieflags
Checking cookie : PHPSESSID
HttpOnly flag is not set in the cookie : PHPSESSID
Secure flag is not set in the cookie : PHPSESSID
Checking cookie : security
HttpOnly flag is not set in the cookie : security
Secure flag is not set in the cookie : security
[*] Launching module http_only
[*] 1 pages were previously attacked and will be skipped
[*] Launching module file
[*] 1 pages were previously attacked and will be skipped
[*] Launching module sql
[*] 1 pages were previously attacked and will be skipped
[*] Launching module xss
[*] 1 pages were previously attacked and will be skipped
[*] Launching module csrf
[*] 1 pages were previously attacked and will be skipped
[*] Asking endpoint URL: https://wapiti.org/c/steffanopol-1939jh for results, please wait...
[!] Unable to request endpoint URL: https://wapiti.org/
[*] Launching module redirect
[*] Launching module blindsql
[*] 1 pages were previously attacked and will be skipped
[*] Launching module parametxss
Report
A report has been generated in the file /home/kali/.wapiti/generated_report
from /home/kali/.wapiti/generated_report/192.168.56.101_w3l3k4z7_001.html with a browser to see this report.
```

Figure 5: Wapiti scanning output after tool use (Kali).

Wapiti vulnerability report
Target: <http://192.168.56.103/dvwa/>
Date of the scan: Mon, 10 Mar 2025 00:14:57 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	0
Blind SQL Injection	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	2
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Htaccess Bypass	0
HTTP Secure Headers	8
HttpOnly Flag cookie	4
Open Redirect	0
Secure Flag cookie	4
SQL Injection	0
Server Side Request Forgery	0
Cross Site Scripting	0
XML External Entity	0
Internal Server Error	0
Resource consumption	0
Fingerprint web technology	0



Figure 6: Wapiti report output in browser.

5.4 Verification of Findings

To verify the vulnerabilities reported by Wapiti, the following actions were taken:

- **Content Security Policy (CSP):**

- The response headers were checked using the command:

```
curl -I http://192.168.56.103/dvwa/
```

The response did not include the `Content-Security-Policy` header.

- **HTTP Secure Headers:**

- The security headers were verified using the same `curl -I` command, and none of the missing headers (`X-Frame-Options`, `X-XSS-Protection`, `X-Content-Type-Options`, `Strict-Transport-Security`) were found in the response.

- **HttpOnly Flag Cookie:**

- The cookies were inspected using the browser's developer tools and the `HttpOnly` flag was confirmed to be missing from the `PHPSESSID` and `security` cookies.

6 Results and Discussion

This section discusses the key findings from the Nikto and Wapiti scans, the implications of these results, and proposes potential fixes.

Nikto Testing Results:

- **Outdated Software:** Apache 2.2.8 and PHP 5.2.4 are outdated, exposing the server to known vulnerabilities.
- **Missing HTTP Headers:** Lack of *X-Frame-Options* and *X-Content-Type-Options* increases vulnerability to click-jacking and MIME type spoofing.

Proposed Fixes:

- **Update Software:** Upgrade Apache and PHP to the latest supported versions.
- **Configure Security Headers:** Add necessary headers such as *X-Frame-Options* and *X-Content-Type-Options*.

Wapiti Testing Results:

- **Missing CSP:** No Content Security Policy, exposing the web app to XSS and injection attacks.
- **Missing HTTP Headers:** Missing *X-Frame-Options*, *X-XSS-Protection*, and *Strict-Transport-Security*, which exposes to multiple attack vectors.
- **HttpOnly Flag Missing:** Missing *HttpOnly* flag on session cookies, increasing risk of session hijacking.

Proposed Fixes:

- **Implement CSP:** Add a restrictive Content Security Policy header to mitigate XSS.
- **Set HttpOnly Flag:** Ensure cookies have the *HttpOnly* flag to protect session data.
- **Configure Secure Headers:** Implement missing headers like *X-Frame-Options* and *Strict-Transport-Security*.

Comparison of Nikto and Wapiti Results: Both tools identified critical vulnerabilities in their respective areas. Nikto focused on server-side issues like outdated software and insecure configurations, while Wapiti highlighted application-level flaws such as the lack of a Content Security Policy and insecure cookies. Together, they provide a comprehensive assessment of the server and web application security, showcasing the need for regular updates and secure configuration practices. I have linked/referenced multiple sources describing the criticality of these errors, but needless to say, it is crucial to have tools like Nikto and Wapiti to mitigate this risk.

7 Conclusion

This report has explored the benefits and use cases for Nikto and Wapiti, and I hope that it is evident how important tools like this will be in the future. Used in combination, a web-server scanner and a web-application scanner are critical tools that allow us to find vulnerabilities a human might not immediately catch. If we utilize tools like this in the correct way, (an isolated environment, proper tool usage) we could make the web a much safer place, overall. Mitigating risk on the web is the future going forward, and I hope I have done an adequate job showing just how important it is.

References

- [1] Wapiti web application vulnerability scanner documentation, 2024. Accessed: 2025-03-09. URL: <https://wapiti.sourceforge.io/>.
- [2] Dataspace Academy. Exploring the power of the nikto tool in web security, 2024. Accessed: 2025-03-09. URL: <https://dataspaceacademy.com/blog/exploring-the-power-of-the-nikto-tool-in-web-security>.
- [3] K. G. Coffman and A. M. Odlyzko. Internet growth: Is there a "moore's law" for data traffic?, 2001. Accessed: 2025-03-09. URL: <https://www-users.cse.umn.edu/~odlyzko/doc/internet.moore.pdf>.
- [4] The Apache Software Foundation. Apache http server 2.2.x vulnerabilities, 2025. Accessed: 2025-03-09. URL: https://httpd.apache.org/security/vulnerabilities_22.html.
- [5] Hackviser. Nikto: Web server scanner, 2025. Accessed: 2025-03-09. URL: <https://hackviser.com/tactics/tools/nikto#:~:text=Vulnerability%20Scanning%3A%20Nikto%20is%20a,could%20be%20exploited%20by%20hackers>.
- [6] IbisWorld. Internet traffic volume, 2024. Accessed: 2025-03-09. URL: <https://www.ibisworld.com/us/bed/internet-traffic-volume/88089/>.
- [7] David Lodge. Nikto web scanner documentation, 2021. Accessed: 2025-03-09. URL: <https://github.com/sullo/nikto>.
- [8] Tomasz Andrzej Nidecki. Cookie hijacking: Understanding the risks and how to prevent it, 2020. Accessed: 2025-03-09. URL: <https://www.invicti.com/learn/cookie-hijacking/>.
- [9] SecureFlag. Incorrect content security policy vulnerability, 2025. Accessed: 2025-03-09. URL: https://knowledge-base.secureflag.com/vulnerabilities/security_misconfiguration/incorrect_content_security_policy_vulnerability.html.
- [10] Shivam Tahalani. Nikto penetration testing, 2020. Accessed: 2025-03-09. URL: <https://hackerman007.medium.com/nikto-penetration-testing-2b06cbdd3e27>.