

# **Lab 3: Detect Web Application Vulnerabilities using Various Web Application Security Tools**

## **Lab Scenario**

When talking about web applications, organizations consider security to be a critical component, because web applications are a major source of attacks. Attackers try various application-level attacks to compromise the security of web applications to commit fraud or steal sensitive information.

Web application attacks, launched on port 80/443, go straight through the firewall, past the OS and network-level security, and into the heart of the application, where corporate data resides. Tailor-made web applications are often insufficiently tested, have undiscovered vulnerabilities, and are, therefore, easy prey for hackers.

A professional ethical hacker or pen tester needs to determine whether their organization's website is secure, before hackers download sensitive data, commit crimes using the website as a launchpad, or otherwise endanger the business. There are various web application security assessment tools available to scan, detect, and assess the security and vulnerabilities of web applications. These tools reveal the web application's security posture and are used to find ways to harden security and create robust web applications. These tools automate the process of accurate web-app security assessment, thus enabling cybersecurity staff to protect their business from impending hacker attacks!

The tasks in this lab will assist in discovering the underlying vulnerabilities and flaws in the target web application.

## **Lab Objectives**

- Detect web application vulnerabilities using wapiti web application security scanner

## **Overview of Web Application Security**

Web application security deals with securing websites, web applications, and web services. Web application security includes secure application development, input validation, creating and following security best practices, using WAF Firewall/IDS, and performing regular auditing of a network using web application security tools.

Web Application security tools are automated tools that scan web applications, normally from the outside, to look for security vulnerabilities such as XSS, SQL injection, command injection, path traversal, and insecure server configuration. This category of tools is frequently referred to as Dynamic Application Security Testing (DAST) Tools.

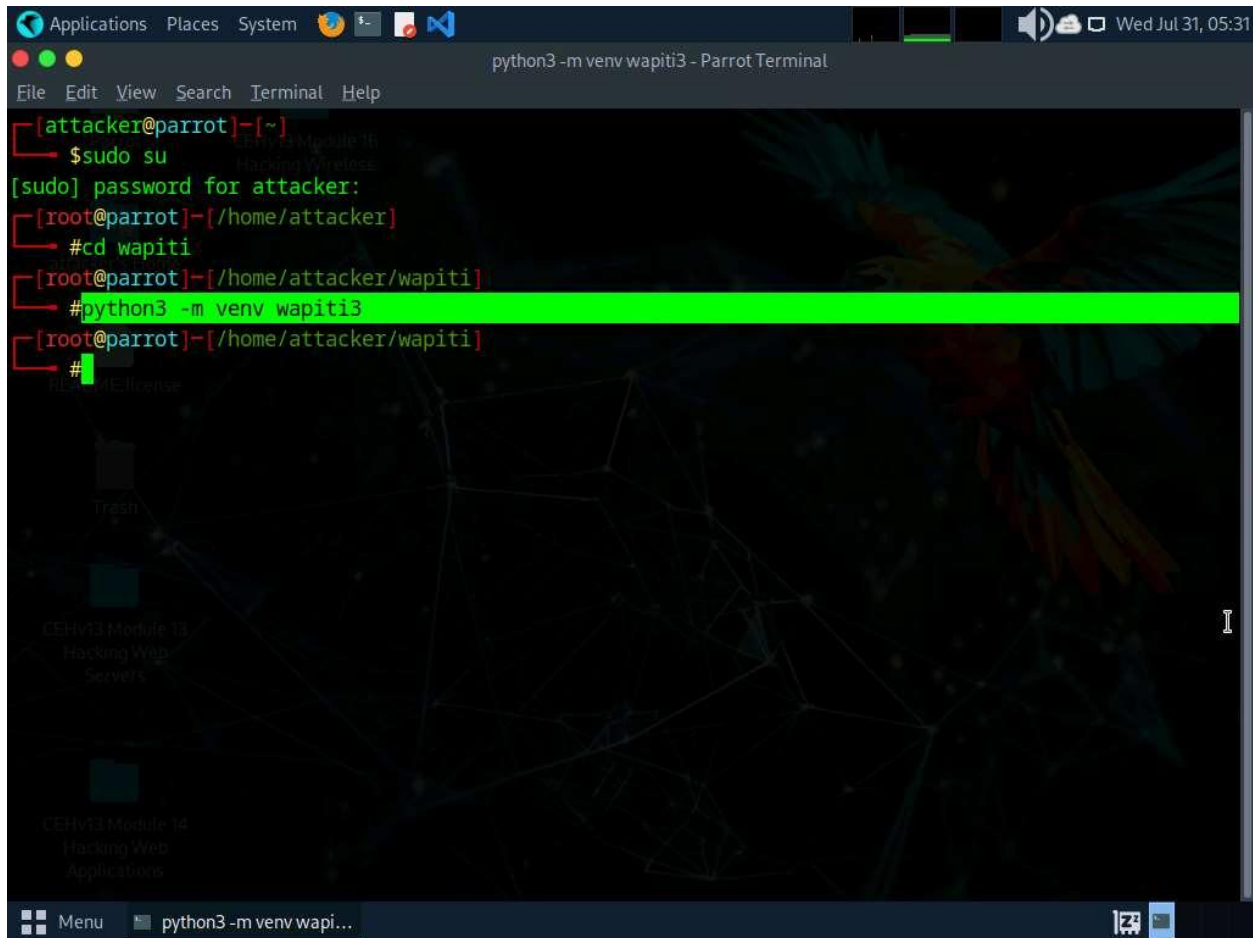
### **Task 1: Detect Web Application Vulnerabilities using Wapiti Web Application Security Scanner**

The Wapiti web-application vulnerability scanner identifies security weaknesses in web applications by crawling websites and performing black-box testing. It detects issues like SQL injections, XSS, and other vulnerabilities.

1. Click [Parrot Security](#) to switch to the **Parrot Security** machine. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

2. In the terminal window run **cd wapiti** command to navigate into wapiti directory and run **python3 -m venv wapiti3** command to create virtual environment in python.



The screenshot shows a terminal window titled "python3 -m venv wapiti3 - Parrot Terminal". The terminal output is as follows:

```
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~# cd wapiti
[root@parrot]~/wapiti# python3 -m venv wapiti3
[root@parrot]~/wapiti#
```

The terminal background features a dark theme with a parrot illustration and a network diagram. The desktop environment includes a menu bar at the top with "Applications", "Places", and "System" menus, and a taskbar at the bottom with a "Menu" button and a window titled "python3 -m venv wapi...".

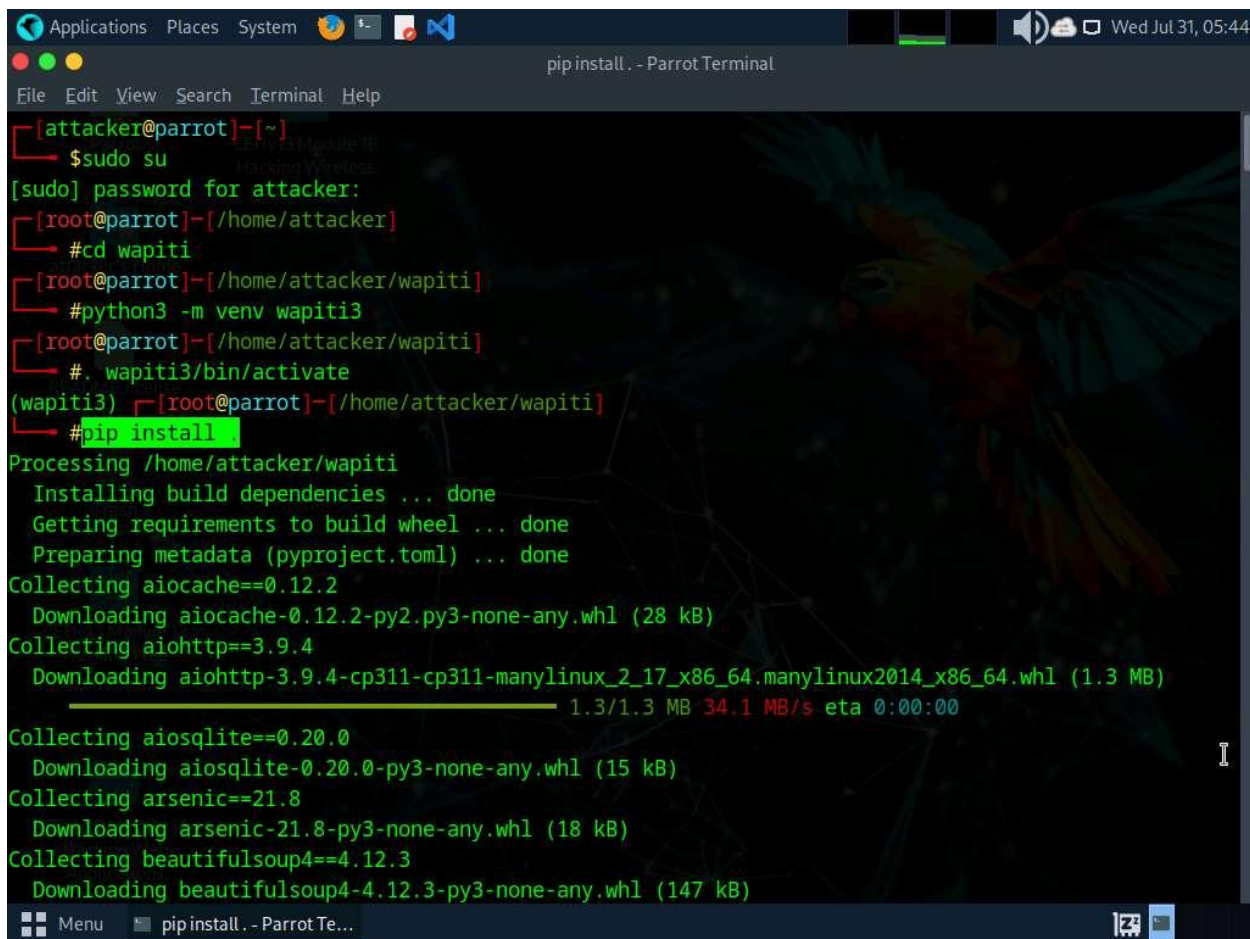
3. Now, run **. wapiti3/bin/activate** command to activate virtual environment.

The screenshot shows a terminal window titled ".wapiti3/bin/activate - Parrot Terminal". The user is logged in as "attacker@parrot" in the home directory. They execute the following commands:

```
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~/home/attacker# cd wapiti
[root@parrot]~/home/attacker/wapiti# python3 -m venv wapiti3
[root@parrot]~/home/attacker/wapiti# . wapiti3/bin/activate
(wapiti3) [root@parrot]~/home/attacker/wapiti#
```

The terminal background features a dark theme with a parrot illustration on the right and a network diagram on the left. The desktop environment includes a menu bar at the top with "Applications", "Places", and "System" menus, and a taskbar at the bottom with a "Menu" button and the active terminal window.

4. Run **`pip install .`** command to install wapiti web application security scanner.



```
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~/home/attacker$ #cd wapiti
[root@parrot]~/home/attacker/wapiti$ #python3 -m venv wapiti3
[root@parrot]~/home/attacker/wapiti$ #. wapiti3/bin/activate
(wapiti3) [root@parrot]~/home/attacker/wapiti$ #pip install
Processing /home/attacker/wapiti
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting aiocache==0.12.2
  Downloading aiocache-0.12.2-py2.py3-none-any.whl (28 kB)
Collecting aiohttp==3.9.4
  Downloading aiohttp-3.9.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
  1.3/1.3 MB 34.1 MB/s eta 0:00:00
Collecting aiosqlite==0.20.0
  Downloading aiosqlite-0.20.0-py3-none-any.whl (15 kB)
Collecting arsenic==21.8
  Downloading arsenic-21.8-py3-none-any.whl (18 kB)
Collecting beautifulsoup4==4.12.3
  Downloading beautifulsoup4-4.12.3-py3-none-any.whl (147 kB)
```

5. After installing the tool run **wapiti -u <https://www.certifiedhacker.com>** command to perform web application security scanning on certifiedhacker.com website.

It takes approximately 10 minutes for the scan to complete.

The screenshot shows a terminal window with the following content:

```
wapiti -u https://www.certifiedhacker.com - Parrot Terminal
File Edit View Search Terminal Help
(wapiti3) [root@parrot]-[/home/attacker/wapiti]
# wapiti -u https://www.certifiedhacker.com

  _ _ _ _ _
 / / \ \ _ _ _ _ ( _ ) | ( _ ) _ /
 \ \ / \ / _ ' | ' \ | | _ | | _ \
  \ \ / ( _ | | | _ | | | _ | ) |
   \ \ \ _ , | _ / | | \ _ | _ /
      | _ |
      | _ |

Wapiti 3.2.0 (wapiti-scanner.github.io)
[*] Saving scan state, please wait...

[*] Launching module upload

[*] Launching module ssl
Certificate subject: cpcontacts.demo.certifiedhacker.com
Alt. names: autodiscover.certifiedhacker.com, autodiscover.demo.certifiedhacker.com, certifiedhacker.com, cpanel.certifiedhacker.com, cpanel.demo.certifiedhacker.com, cpcalendars.certifiedhacker.com, cpcalendars.certifiedhacker.com, cpcontacts.certifiedhacker.com, cpcontacts.demo.certifiedhacker.com, demo.certifiedhacker.com, mail.certifiedhacker.com, mail.demo.certifiedhacker.com, mail.uyr.fvr.mybluehost.me, uyr.fvr.mybluehost.me, webdisk.certifiedhacker.com, webdisk.demo.certifiedhacker.com, webmail.certifiedhacker.com, webmail.demo.certifiedhacker.com, website-215f0f34.certifiedhacker.com, www.certifiedhacker.com, www.demo.certifiedhacker.com, www.uyr.fvr.mybluehost.me, www.website-215f0f34.certifiedhacker.com
Issuer: R3
Key: RSA 2048 bits
```

```
Applications Places System [Icons] [Volume] [Network] [Battery] [Time] Wed Jul 31, 06:26
cd /root/.wapiti/generated_report/ - Parrot Terminal
File Edit View Search Terminal Help
* TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 strong
* TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 strong
Accepted cipher suites for TLSv1.3:
* TLS_AES_256_GCM_SHA384 strong
* TLS_CHACHA20_POLY1305_SHA256 strong
* TLS_AES_128_GCM_SHA256 strong
[*] Launching module xss
[*] Launching module upload
[*] Launching module csp
CSP is not set
[*] Launching module permanentxss
[*] Generating report...
A report has been generated in the file /root/.wapiti/generated_report
Open /root/.wapiti/generated_report/certifiedhacker.com_07312024_1016.html with a browser to see this
report.
(wapiti3) [root@parrot]-[/home/attacker/wapiti]
#cd /root/.wapiti/generated_report/
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
#
```

7. Run `ls` command to view the contents of the directory. we can see that the `certifiedhacker.com_07312024_1016.html` file is created.

The name of the .html file varies when you perform this lab.



```
Applications Places System [Icons] [Network] [Volume] [Battery] [Clock] Wed Jul 31, 06:29
ls --color=auto - Parrot Terminal
File Edit View Search Terminal Help
Accepted cipher suites for TLSv1.3:
* TLS_AES_256_GCM_SHA384 strong
* TLS_CHACHA20_POLY1305_SHA256 strong
* TLS_AES_128_GCM_SHA256 strong

[*] Launching module xss

[*] Launching module upload

[*] Launching module csp
CSP is not set

[*] Launching module permanentxss

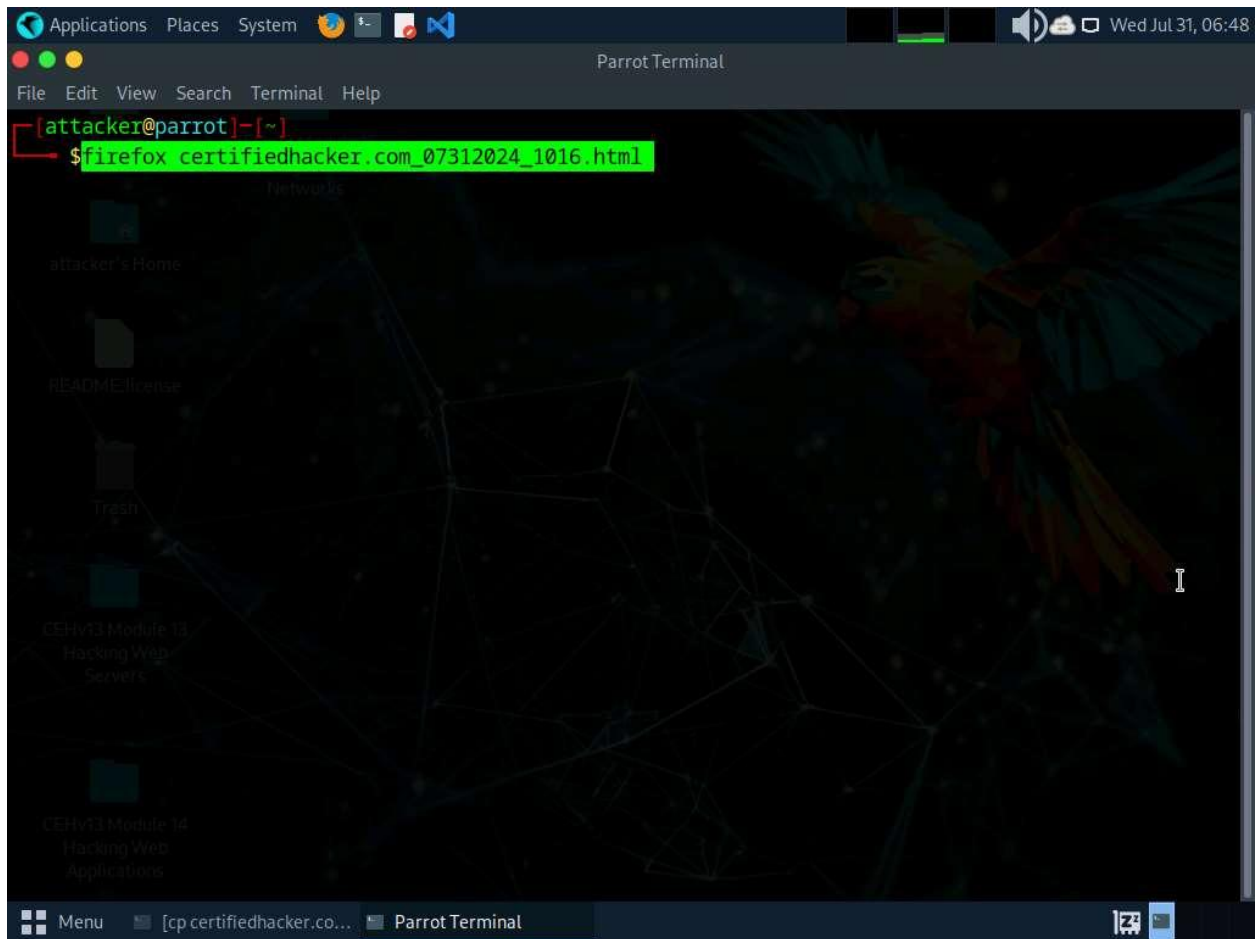
[*] Generating report...
A report has been generated in the file /root/.wapiti/generated_report
Open /root/.wapiti/generated_report/certifiedhacker.com_07312024_1016.html with a browser to see this
report.
(wapiti3) [root@parrot]-[/home/attacker/wapiti]
└─ #cd /root/.wapiti/generated_report/
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
└─ #ls
certifiedhacker.com_07312024_1016.html css js logo_clear.png report.html
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
└─ #
```

8. Run `cp certifiedhacker.com_07312024_1016.html /home/attacker/` command to copy the .html file to `/home/attacker` location.

```
Applications Places System cp certifiedhacker.com_07312024_1016.html /home/attacker/ - Parrot Terminal
File Edit View Search Terminal Help
* TLS_CHACHA20_POLY1305_SHA256 strong
* TLS_AES_128_GCM_SHA256 strong
[*] Launching module xss
[*] Launching module upload
[*] Launching module csp
CSP is not set
[*] Launching module permanentxss
[*] Generating report...
A report has been generated in the file /root/.wapiti/generated_report
Open /root/.wapiti/generated_report/certifiedhacker.com_07312024_1016.html with a browser to see this
report.
(wapiti3) [root@parrot]-[/home/attacker/wapiti]
#cd /root/.wapiti/generated_report/
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
#ls
certifiedhacker.com_07312024_1016.html css js logo_clear.png report.html
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
#cp certifiedhacker.com_07312024_1016.html /home/attacker/
(wapiti3) [root@parrot]-[~/ .wapiti/generated_report]
#
```

9. Open a new terminal and run **firefox certifiedhacker.com\_07312024\_1016.html** command to open the .html file in Firefox browser.





10. Wapiti scan report opens up in Firefox browser, you can analyze the scan result with the discovered vulnerabilities.

Applications Places System Wed Jul 31, 06:50

Wapiti scan report

file:///home/attacker/certifiedhacker.com\_07312024\_1016.html

Import bookmarks... Parrot OS Hack The Box OSINT Services Vuln DB Privacy and Security Learning Resources

## Wapiti vulnerability report

**Target:** <https://certifiedhacker.com/>

Date of the scan: Wed, 31 Jul 2024 10:16:22 +0000. Scope of the scan: folder. Crawled pages: 24

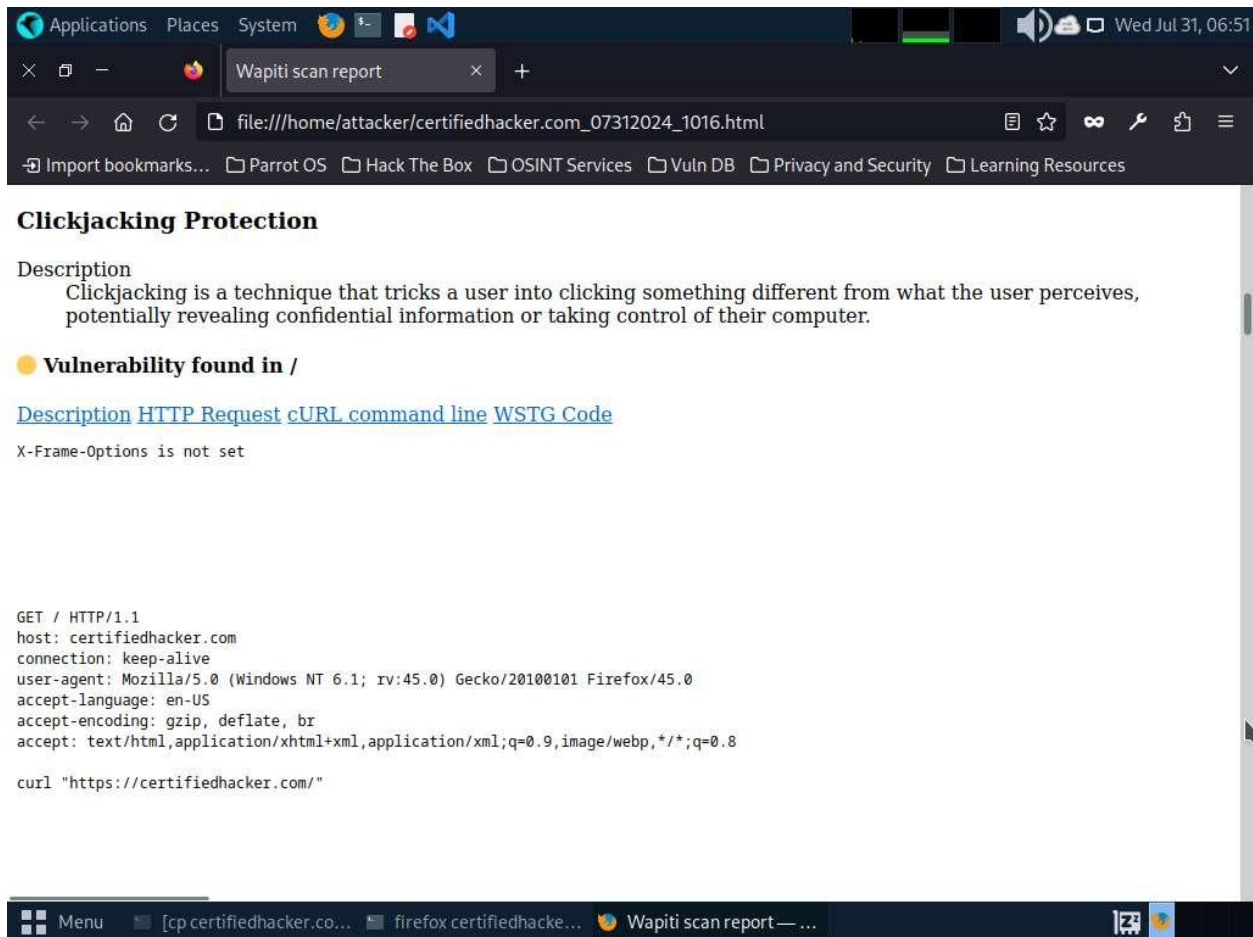
---

### Summary

Category	Number of vulnerabilities found
Backup file	0
Weak credentials	0
CRLF Injection	0
<a href="#">Content Security Policy Configuration</a>	1
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Fingerprint web application framework	0
Fingerprint web server	0
Htaccess Bypass	0
HTML Injection	0
<a href="#">Clickjacking Protection</a>	1
<a href="#">HTTP Strict Transport Security (HSTS)</a>	1
<a href="#">MIME Type Confusion</a>	1
Unvalidated Redirects	0

Menu [cp certifiedhacker.co... firefox certifiedhacke... Wapiti scan report — ...

11. Scroll down to view the detailed information regarding each discovered vulnerability.



12. This concludes the demonstration of discovering vulnerabilities in a target website scanning using wapiti.

13. Close all open windows and document all acquired information.

#### Question 14.3.1.1

In Parrot Security machine use wapiti web application security scanner to detect web application vulnerabilities of <https://www.certifiedhacker.com> web application and generate a .html report. Enter the WSTG code of the Clickjacking Protection vulnerability. (Answer Format: XXXX-X-Xxxxx-Xxxxxxx)