

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.

A screenshot of a Parrot Security terminal window titled "python3 -m venv wapiti3 - Parrot Terminal". The terminal session shows the following commands being run:

```
[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]
#
```

The terminal window is set against a dark background with a parrot-themed wallpaper. The desktop environment includes icons for "CEHv13 Module 13 Hacking Wireless", "CEHv13 Module 14 Hacking Web Servers", and "CEHv13 Module 16 Hacking Wireless". The taskbar at the bottom shows the terminal window is active and displays the command "python3 -m venv wapiti3".

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

The screenshot shows a Parrot OS desktop environment. In the foreground, a terminal window titled ".wapiti3/bin/activate - Parrot Terminal" is open, displaying a root shell session. The user has run several commands to navigate to the wapiti3 directory and activate its virtual environment:

```
[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]
#
```

In the background, a file browser window is visible, showing a folder structure for "CEHv13 Module 13 Hacking Wireless Servers" and "CEHv13 Module 14 Hacking Web Applications". The desktop environment includes a dock with icons for Applications, Places, System, and a terminal.

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

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

```
Applications Places System Terminal Help wapiti -u https://www.certifiedhacker.com - Parrot Terminal  
(wapiti3) [root@parrot]~[~/home/attacker/wapiti]  
#wapiti -u https://www.certifiedhacker.com  
Networks  
/ / \ \ _ _ _ _ _ ( ) |_( )_ /  
\\ \\ \\ \\ _ | ' _ \| | _ | + | _ \  
 \ / / ( _ | | _ ) | | | _ | _ ) |  
 \ \ \ \ _ , | . / | _ \ _ | _ /  
 | _ |  
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, cp.calendars.demo.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  
wapiti -u https://www.certifiedhacker.com
```

- Now, in the terminal run `cd /root/.wapiti/generated_report/` to navigate to generated_report directory.

The screenshot shows a terminal window on a Parrot OS desktop environment. The terminal window has a dark blue header bar with icons for Applications, Places, System, and a few others. The title bar says "cd /root/.wapiti/generated_report/ - ParrotTerminal". The main area of the terminal shows the output of a Wapiti command:

```
* 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]~[~/wapiti/generated_report]
(wapiti3) #cd /root/.wapiti/generated_report/
(wapiti3) [root@parrot]~[~/wapiti/generated_report]
(wapiti3) #
```

The terminal window has a green status bar at the bottom with the text "cd /root/.wapiti/gener...".

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

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

The screenshot shows a terminal window titled "ls --color=auto - Parrot Terminal". The terminal displays the output of the Wapiti web application penetration testing tool. The output includes:

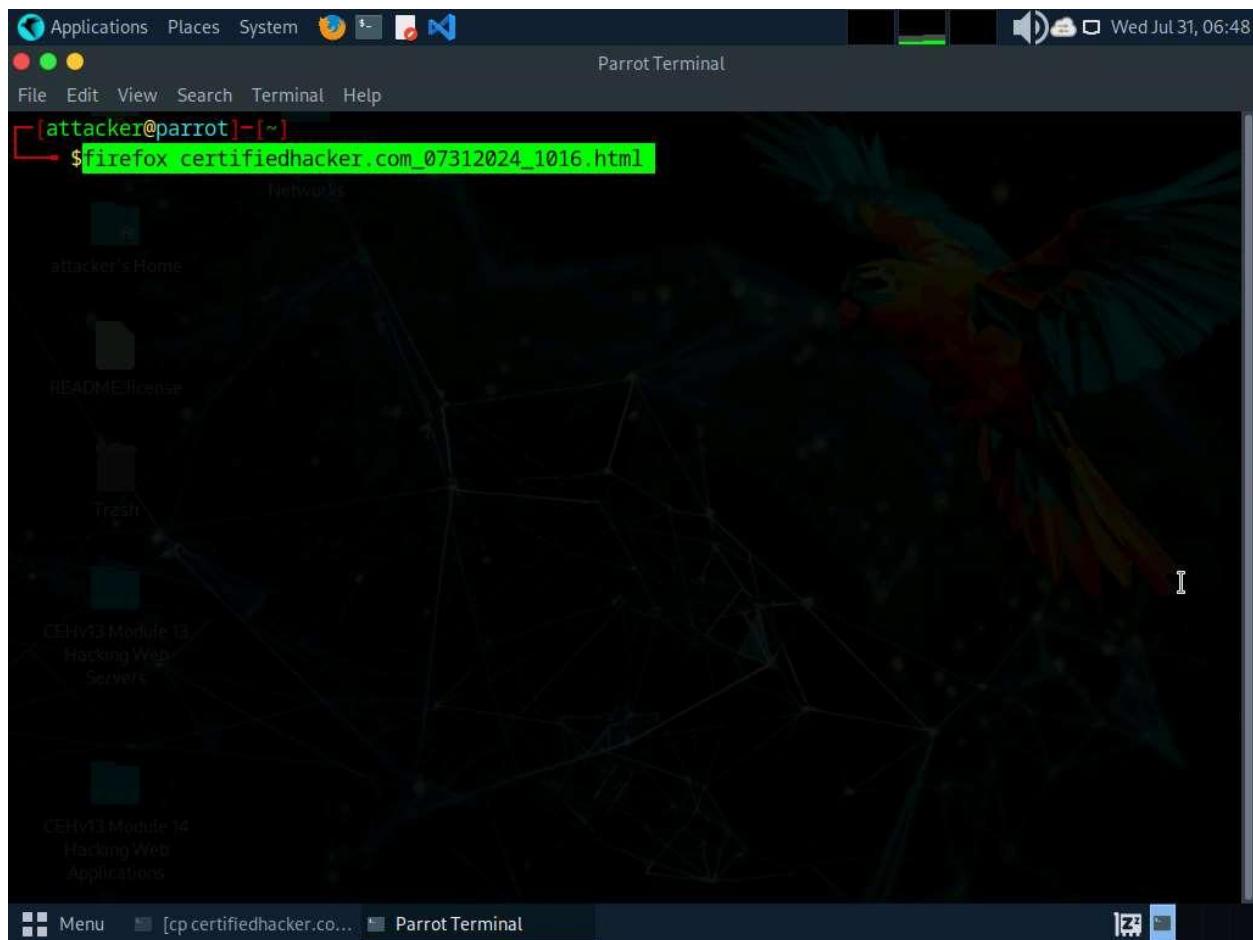
- Accepted cipher suites for TLSv1.3:
 - * TLS_AES_256_GCM_SHA384 strong
 - * TLS_CHACHA20_POLY1305_SHA256 strong
 - * TLS_AES_128_GCM_SHA256 strong
- [*] Launching module xss
- [*] Launching module upload
- [*] Launching module csp
CSP is not set
- [*] Launching module permanentxss
- [*] Generating 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]~/.wapiti/generated_report/
└─# 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_xxxxxxxxxx_xxxx.html /home/attacker/** command to copy the .html file to **/home/attacker** location.

The terminal window shows the following session:

```
* TLS_CHACHA20_POLY1305_SHA256 strong
* TLS_AES_128_GCM_SHA256 strong
[*] Launching module xss
[*] Launching module upload
[*] Launching module csp
CSP is not set
README license
[*] 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]
└─#
# Applications
Menu cp certifiedhacker.co...
```

9. Open a new terminal and run **firefox certifiedhacker.com_xxxxxxxxxx_xxxx.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.

Wapiti vulnerability report

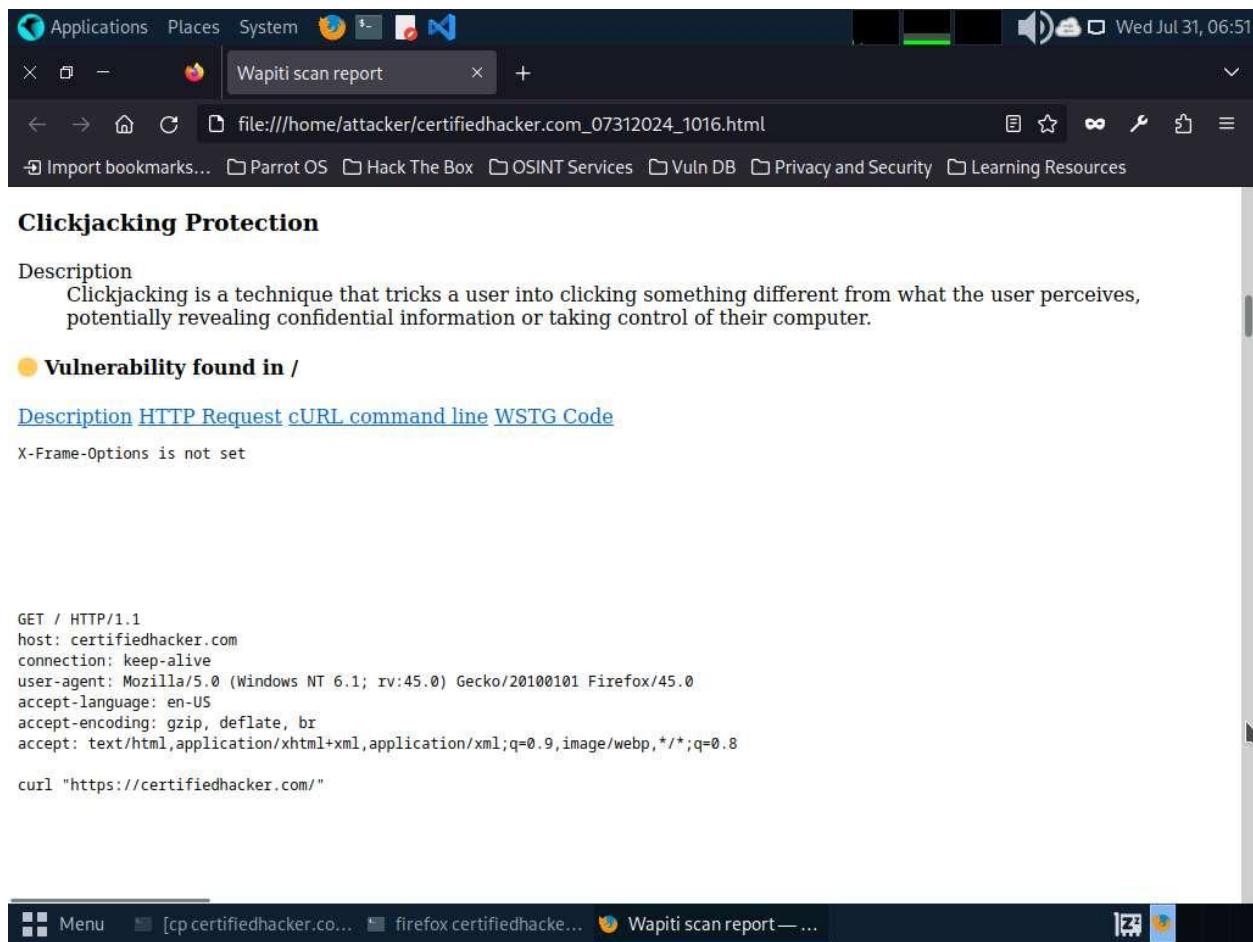
Target: <https://certifiedhacker.com/>

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
Content Security Policy Configuration	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
Clickjacking Protection	1
HTTP Strict Transport Security (HSTS)	1
MIME Type Confusion	1
Other	0

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



Clickjacking Protection

Description
Clickjacking is a technique that tricks a user into clicking something different from what the user perceives, potentially revealing confidential information or taking control of their computer.

● Vulnerability found in /

[Description](#) [HTTP Request](#) [cURL command line](#) [WSTG Code](#)

X-Frame-Options is not set

```
GET / HTTP/1.1
host: certifiedhacker.com
connection: keep-alive
user-agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0
accept-language: en-US
accept-encoding: gzip, deflate, br
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

curl "https://certifiedhacker.com/"
```

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-Xxxxxx-Xxxxxxx)