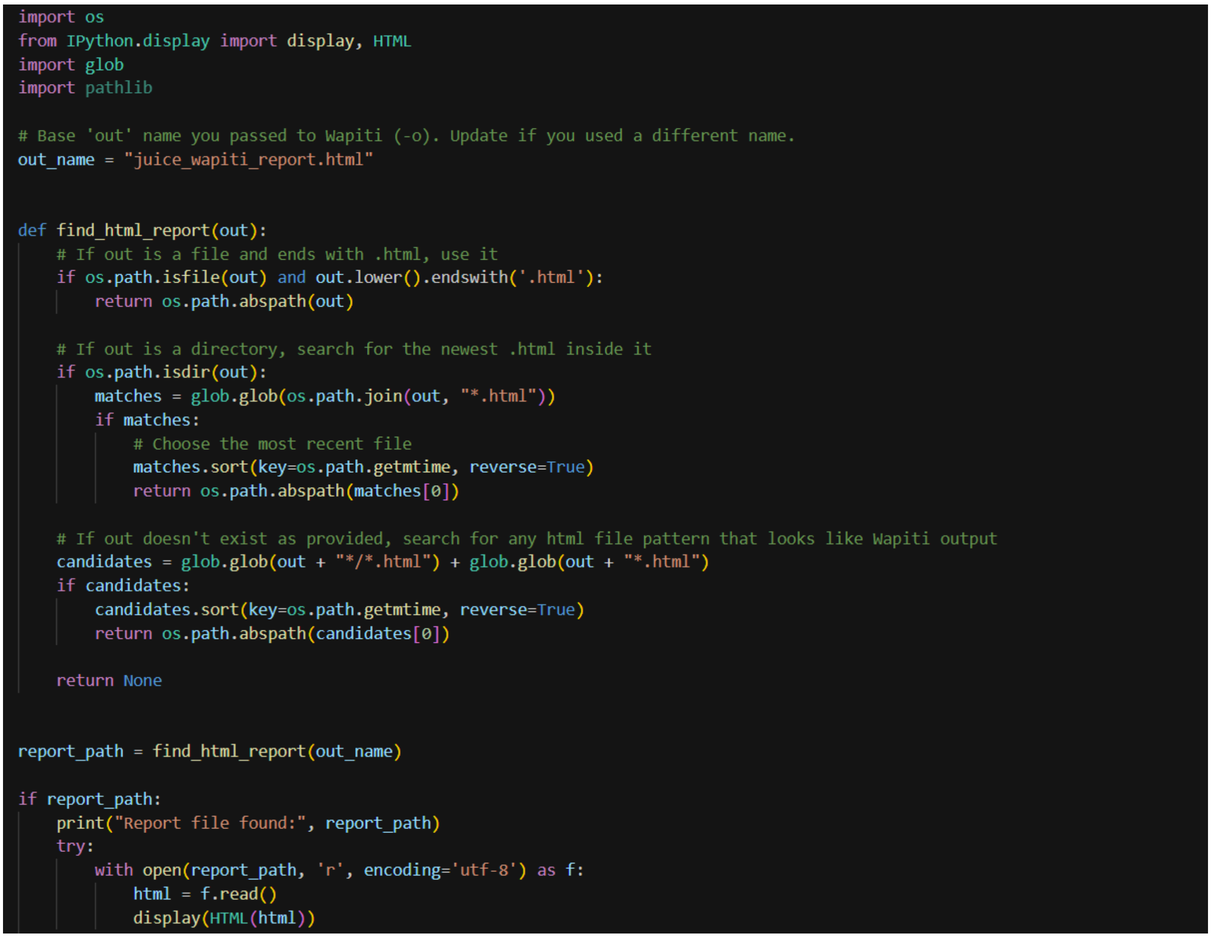
Lab 5

The aim of this lab was to understand **how security professionals identify vulnerabilities in web applications using automated scanners**. Instead of manually testing every input, a vulnerability scanner sends thousands of crafted requests and analyses the responses to uncover weaknesses.

In this lab, I used:

- Wapiti - a black-box web vulnerability scanner
- Vulnerable training websites such as Google Gruyere and OWASP Juice Shop

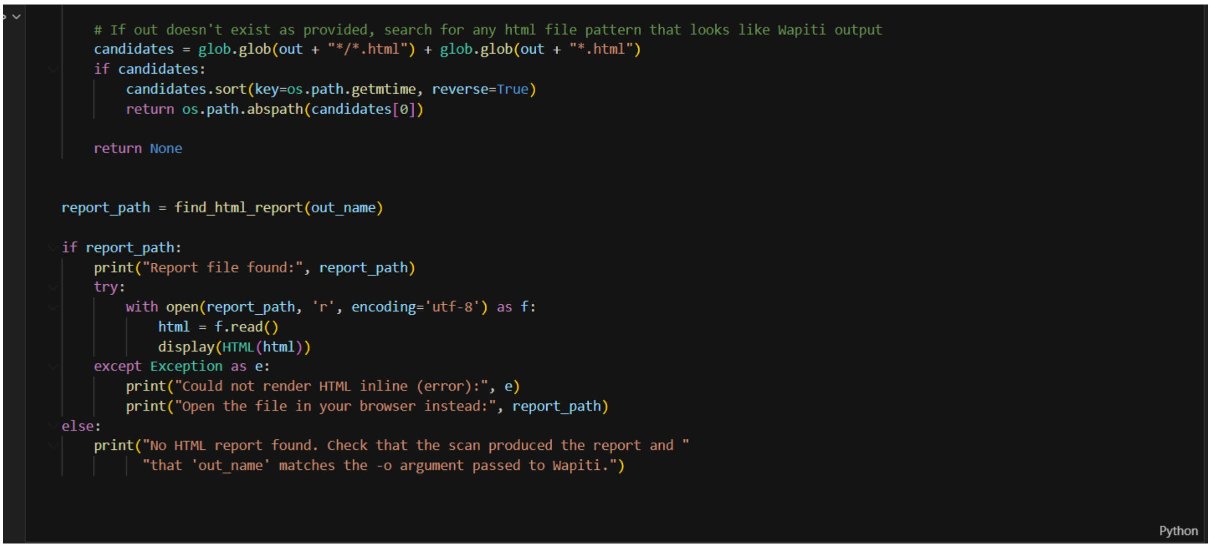
The goal was to **safely simulate real-world penetration testing** and learn how web flaws are discovered before attackers exploit them.



```python
import os
from IPython.display import display, HTML
import glob
import pathlib

# Base 'out' name you passed to Wapiti (-o). Update if you used a different name.
out_name = "juice_wapiti_report.html"


def find_html_report(out):
    # If out is a file and ends with .html, use it
    if os.path.isfile(out) and out.lower().endswith('.html'):
        return os.path.abspath(out)

    # If out is a directory, search for the newest .html inside it
    if os.path.isdir(out):
        matches = glob.glob(os.path.join(out, "*.html"))
        if matches:
            # Choose the most recent file
            matches.sort(key=os.path.getmtime, reverse=True)
            return os.path.abspath(matches[0])

    # If out doesn't exist as provided, search for any html file pattern that looks like Wapiti output
    candidates = glob.glob(out + "*/*.html") + glob.glob(out + "*.html")
    if candidates:
        candidates.sort(key=os.path.getmtime, reverse=True)
        return os.path.abspath(candidates[0])

    return None


report_path = find_html_report(out_name)

if report_path:
    print("Report file found:", report_path)
    try:
        with open(report_path, 'r', encoding='utf-8') as f:
            html = f.read()
            display(HTML(html))
```

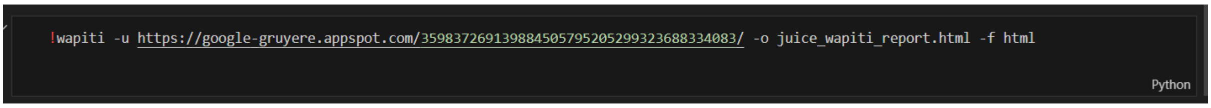
Helps set out the template for the HTML report

Lab 5

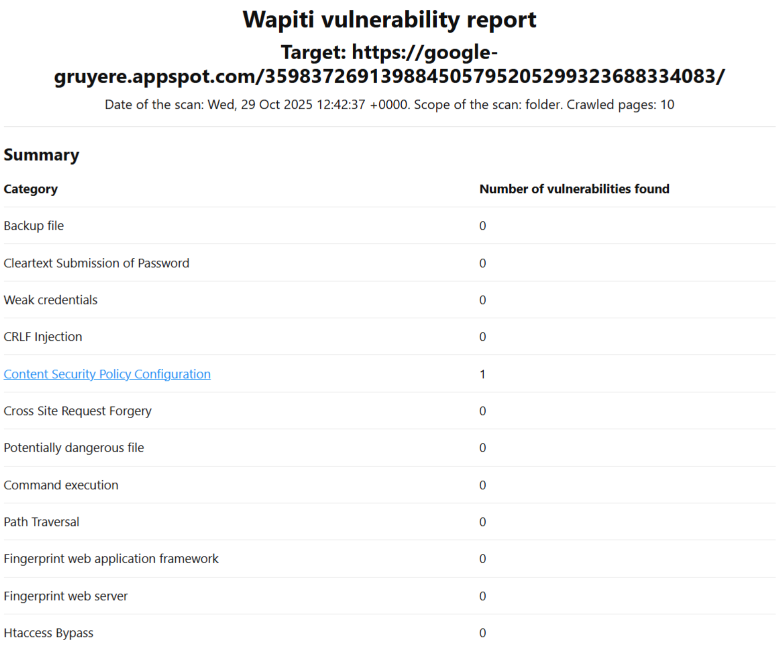


```python
    # If out doesn't exist as provided, search for any html file pattern that looks like Wapiti output
    candidates = glob.glob(out + "*/*.html") + glob.glob(out + "*.html")
    if candidates:
        candidates.sort(key=os.path.getmtime, reverse=True)
        return os.path.abspath(candidates[0])

    return None


report_path = find_html_report(out_name)

if report_path:
    print("Report file found:", report_path)
    try:
        with open(report_path, 'r', encoding='utf-8') as f:
            html = f.read()
            display(HTML(html))
    except Exception as e:
        print("Could not render HTML inline (error):", e)
        print("Open the file in your browser instead:", report_path)
else:
    print("No HTML report found. Check that the scan produced the report and "
          "that 'out_name' matches the -o argument passed to Wapiti.")
```

If statement if the report is generated Else statement in case the report cannot be generated



```
!wapiti -u https://google-gruyere.appspot.com/359837269139884505795205299323688334083/ -o juice_wapiti_report.html -f html
```



Wapiti vulnerability report

Target: https://google-gruyere.appspot.com/359837269139884505795205299323688334083/

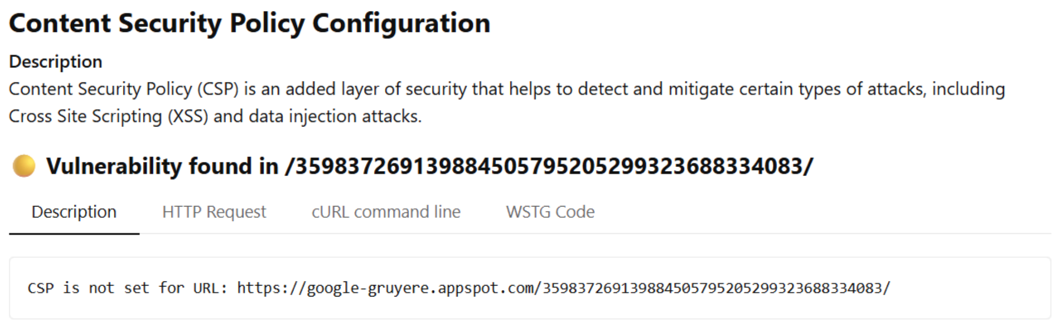
Date of the scan: Wed, 29 Oct 2025 12:42:37 +0000. Scope of the scan: folder. Crawled pages: 10

Summary

Category	Number of vulnerabilities found
Backup file	0
Cleartext Submission of Password	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

Snippet of the vulnerability report summary

Lab 5



Content Security Policy Configuration

Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.

● Vulnerability found in /359837269139884505795205299323688334083/

```
CSP is not set for URL: https://google-gruyere.appspot.com/359837269139884505795205299323688334083/
```

An example of a vulnerability