# Widow-Web: An Enhanced Security-Focused Web Crawler

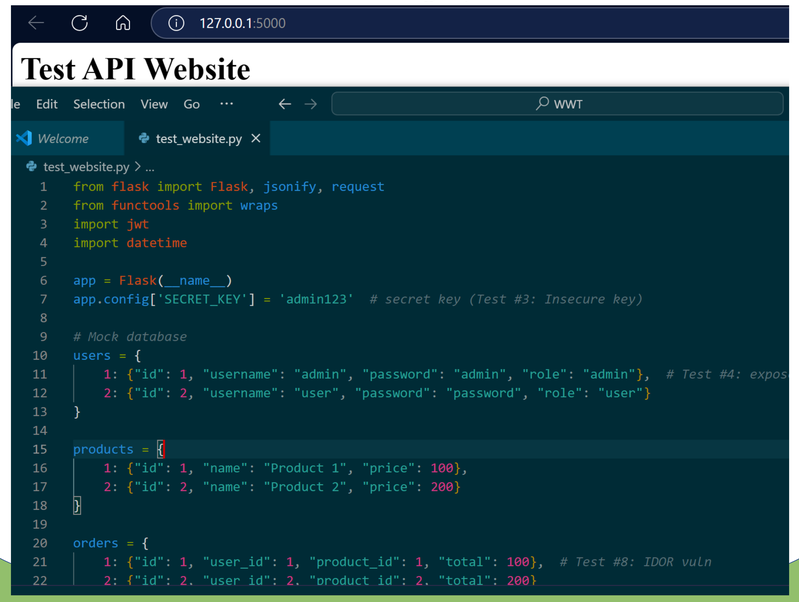
Cassidy Jensen
Computer Science Department, Utah Valley University
Faculty Advisor: Dr. Sayeed Sajal

## Introduction

Web application security remains a challenge as vulnerabilities continue to grow. As organizations increasingly rely on APIs, security risks also grow, including injection flaws, insecure APIs, and broken authentication. Manual audits are not enough with the complexity of modern web applications, creating a need for automated security tools. This project addresses that gap by developing a security-focused web crawler designed to detect vulnerabilities. By integrating API vulnerabilities along with other common web vulnerabilities, the tool enhances the efficiency of security assessments, helping developers identify risks early in the development process.

## The Test Website

Due to the sensitive information and legal restrictions around testing on public sites, I first created a locally hosted test website designed to evaluate my crawler's effectiveness. The website contains a variety of vulnerabilities for the crawler to detect. One API-specific vulnerability included an API key that did not require credentials, allowing unauthorized access. Additional vulnerabilities include exposed user data, no rate limiting allowing potential API abuse, and other security issues. This controlled environment allowed me to thoroughly test the crawler.



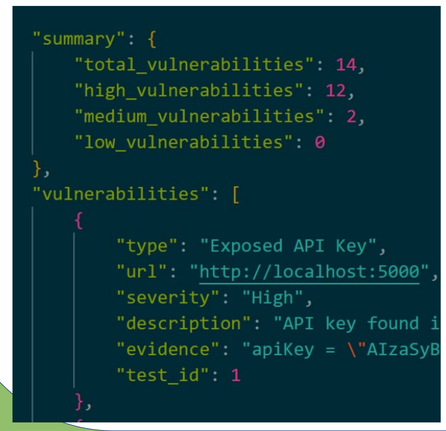

## Background Research and Tools

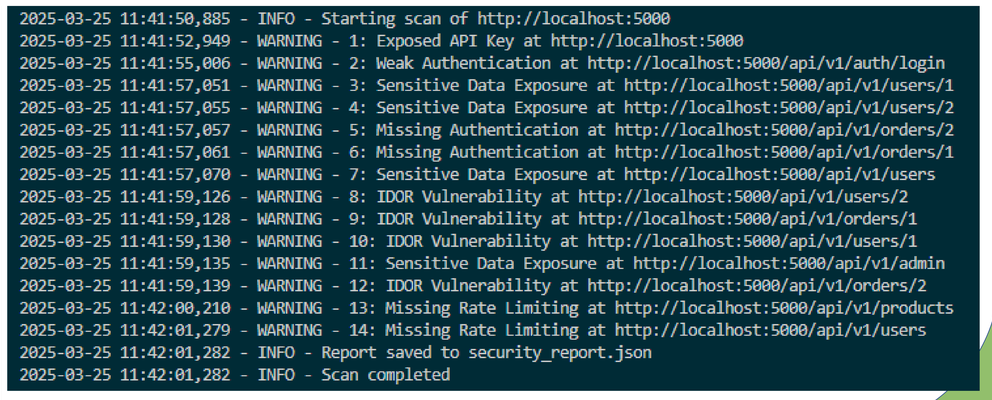
Importance of Web Security: In an age where internet accessibility is widespread, neglecting web security can lead to data breaches, loss of user trust, financial loss, and legal consequences.
Comparison of Web Crawlers: The most widely used web crawler for security testing is Burp Suite by PortSwigger, a comprehensive web and API vulnerability assessments. However, it is costly, priced at nearly four hundred dollars per year, with the free version lacking essential features. An alternative is OWASP ZAP (Zed Attack Proxy), a free and frequently updated tool that is scalable and reliable, though it suffers from outdated documentation and limited reporting capabilities.

## The Crawler

The crawler's core structure was built around a main class named WidowsWeb, designed to be modular.
The main function served as the entry point to run the crawler, efficiently calling the main class to initiate the scan.
A JSON report generation function was implemented early on to create a detailed log of detected vulnerabilities, including the type of vulnerability and affected endpoint.
Shown is a successful scan that produces a JSON report and terminal information of vulnerabilities found.

- setup_logging()
- scan_url(base_url)
- check_exposed_secrets(url)
- test_authentication(base_url) -> str
- test_endpoints(base_url, token)
- test_endpoint_access(url, headers)
- test_for_idor(url, headers)
- test_rate_limiting(base_url)
- add_vulnerability(vulnerability)
- save_report(filename='security_report.json')




## References

Waheed, A. (2024, November 29). *API security guide: Threats, solutions & tools*. Apidog An integrated platform for API design, debugging, development, mock, and testing. https://apidog.com/blog/api-security-threats-solution-tools/

*Requirements for API scanning*. PortSwigger. (n.d.). https://portswigger.net/burp/documentation/scanner/api-scanning-reqs