International Journal of Science, Strategic Management and Technology
Volume **02** Issue **05** May-**2026** | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Vulnerability Scanner using Python

Madhumitha N. B.Sc.
Computer Science
VISTAS, Chennai

Ajmal Khan S
B.Sc. Computer Science VISTAS,
Chennai

Dr. Sheela K., M.Sc., Ph.D.
Assistant Professor, Dept. of CS & IT VISTAS, Chennai

https://doi.org/10.55041/ijsmt.v2i5.147

Cite this Article: Madhumitha, & S, A. K. (2026). Vulnerability Scanner using Python. International Journal of Science, Strategic Management and Technology, *02*(05). https://doi.org/10.55041/ijsmt.v2i5.147

ABSTRACT

This paper presents a vulnerability scanner implemented in Python to identify security weaknesses across networked systems and web applications. The scanner integrates multiple modules to perform comprehensive security assessment, including network discovery, service version detection, port scanning, and web vulnerability analysis. A modular architecture enables easy extension with new plugins for common protocols (HTTP, SSH, FTP, SMB) and data sources such as CVE databases and vendor advisories. The tool emphasizes security best practices including least-privilege operation, secure credential handling, and audit logging. Output is generated in both human-readable formats and machine-friendly JSON, with options to export reports to HTML or CSV for integration with CI/CD pipelines. By providing an extensible, Python-based solution, this scanner enables security teams to efficiently identify, prioritize, and remediate vulnerabilities across heterogeneous infrastructures.

1. INTRODUCTION

In the modern digital era, cybersecurity has become one of the most critical aspects of information technology. With the rapid growth of internet-based services, cloud computing, and interconnected systems, the risk of cyber threats has increased significantly. Organizations, governments, and individuals rely heavily on digital platforms for communication, data storage, and business operations. However, this reliance also exposes systems to various security vulnerabilities that can be exploited by malicious actors.

Cybersecurity refers to the practice of protecting systems, networks, and data from unauthorized access, attacks, and damage. It involves implementing measures to safeguard information and ensure the confidentiality, integrity, and availability of data. The increasing number of cyberattacks—including hacking, phishing, malware, and ransomware—highlights the importance of strong security mechanisms.

1.1 Overview of Vulnerability Scanning

Vulnerability scanning is a technique used to identify security weaknesses in systems, networks, and applications. It involves scanning a target system to detect open ports, running services, misconfigurations, and known vulnerabilities.

The primary goal of vulnerability scanning is to provide a proactive approach to security by identifying potential threats before they can be exploited.

A vulnerability scanner works by sending requests to the target system and analyzing the responses. It compares the results with a database of known vulnerabilities to identify potential risks. The process typically includes port scanning, service detection, and vulnerability analysis. Vulnerability scanners can be classified into different types—network scanners, web application scanners, and host-based scanners—each serving a distinct purpose.

1.2 Problem Statement

Despite the availability of various security tools, many organizations and individuals still face challenges in detecting vulnerabilities effectively. Existing tools are often complex, expensive, and require advanced technical knowledge, limiting their accessibility especially for small organizations and students. Manual testing methods are inefficient and cannot keep up with the increasing number of threats.

The lack of integration in existing solutions is also a significant problem. Different tools are often required for different tasks such as port scanning, service detection, and web vulnerability analysis. Managing multiple tools increases complexity and reduces efficiency. There is, therefore, a pressing need for a simple, cost-effective, and automated solution that can perform comprehensive vulnerability scanning within a single platform.

1.3 Aim and Objectives

The primary aim of this project is to design and develop an Automated Vulnerability Scanner using Python that can efficiently identify security weaknesses in computer systems, networks, and web applications. The system is intended to provide a reliable, cost-effective, and user-friendly solution that bridges the gap between expensive commercial tools and the needs of learners and small-scale practitioners.

Specific objectives include: (i) implementing automated port scanning functionality; (ii) performing service detection on open ports; (iii) incorporating web vulnerability detection for SQL injection and cross-site scripting;
(iv) generating detailed, structured reports with severity ratings; and (v) providing a user-friendly interface accessible to non-experts.

2. LITERATURE REVIEW

The literature review provides an understanding of existing research, tools, and techniques related to the problem domain. This section studies existing vulnerability scanning tools, methodologies, and research contributions in the field of cybersecurity, helping to identify gaps in existing systems and providing a foundation for developing an improved solution.

2.1 Types of Vulnerability Scanners

Vulnerability scanners can be categorized based on their functionality and scope. Network-based scanners focus on identifying vulnerabilities in network infrastructure, while host-based scanners analyze individual systems. Web application scanners are designed to detect vulnerabilities in web applications, such as SQL injection and cross-site scripting (XSS). Database scanners focus on identifying weaknesses in database systems. Modern tools often combine multiple functionalities to provide comprehensive analysis.

2.2 Existing Tools and Technologies

Several tools are available for vulnerability scanning. Popular commercial tools such as Nessus, Qualys, and Rapid7 InsightVM provide powerful scanning capabilities but require expensive licenses and specialized knowledge. Open-source alternatives such as OpenVAS and Nikto are more accessible, but may lack advanced features or user-friendly interfaces. The complexity and cost of these tools limit their accessibility for beginners and small organizations.

2.3 Python in Cybersecurity

Python has become one of the most popular programming languages in cybersecurity due to its simplicity and versatility. It provides a wide range of libraries for network communication (socket, scapy), data analysis, and web interaction (requests, BeautifulSoup). Python is widely used for developing security tools including vulnerability scanners, penetration testing scripts, and automation frameworks. Its ease of use makes it suitable for both beginners and seasoned professionals, making it the ideal choice for this project.

2.4 Research Gaps

The literature review reveals several notable gaps. Many tools lack integration, requiring users to switch between different platforms. Some tools are too complex for beginners, while others are too limited in functionality. There is a clear need for cost-effective solutions that can be used by students and small organizations. Additionally, many existing systems do not provide clear and user-friendly reports. These gaps justify the development of an integrated, automated, and user-friendly vulnerability scanning system.

3. SYSTEM ANALYSIS

3.1 Existing System

The existing approach to vulnerability detection relies on a combination of manual methods and standalone tools. Security professionals use separate tools for port scanning, service detection, and web vulnerability analysis. While these tools are powerful, they are not integrated into a single platform, making the process complex and time-consuming. Commercial vulnerability scanners provide advanced features but are expensive and require specialized knowledge, limiting their accessibility to students and small organizations.

3.2 Drawbacks of Existing System

Key limitations of the existing system include: (i) lack of integration—users must rely on multiple tools, increasing complexity; (ii) high cost—many advanced tools require expensive licenses; (iii) complexity—most tools are designed for experienced professionals and are not suitable for beginners; and (iv) time consumption—manual testing methods are inefficient for large and complex systems. These limitations collectively highlight the need for an automated and integrated solution.

3.3 Proposed System

The proposed system is an Automated Vulnerability Scanner using Python that integrates multiple functionalities into a single platform. It performs port scanning, service detection, web vulnerability analysis, and report generation in an automated manner. The system provides a user-friendly interface that allows users to input a target IP address or URL and initiate the scanning process with minimal effort. Automation ensures scanning is performed quickly and consistently, reducing the chances of human error.

4. SYSTEM REQUIREMENTS

4.1 Functional Requirements

The system must allow users to input a target IP address or URL. Once input is provided, the system initiates scanning automatically. The port scanning module identifies open ports and provides relevant information. The service detection module determines services running on open ports. A web vulnerability scanning module checks for SQL injection and XSS. The system generates detailed reports including vulnerability severity levels and recommended mitigations.

4.2 Non-Functional Requirements

Performance: the system performs scanning operations efficiently within a reasonable time frame. Usability: a simple

interface allows users to perform scans without advanced technical knowledge. Reliability: the system provides accurate results and handles errors gracefully. Security: scanning operations are performed safely without causing harm to the target system.

Hardware Requirements	Software Requirements
• Minimum 4 GB RAM • Modern multi-core processor • Stable network connection • Adequate disk storage for reports	• Python 3.x (primary language) • Libraries: socket, requests • IDE: VS Code / PyCharm • OS: Windows / Linux / macOS

5. SYSTEM DESIGN

The design of the vulnerability scanner follows a modular approach, where the system is divided into smaller components, each performing a specific function. This approach improves maintainability, scalability, and flexibility. The system follows a layered architecture: the input layer collects user data, the processing layer performs scanning and analysis, and the output layer generates reports.

5.1 System Architecture

The vulnerability scanner adopts a three-tier layered architecture. The Presentation Layer handles user input and output. The Business Logic Layer contains the core scanning and analysis engines. The Data Layer manages result storage and report generation. This separation of concerns makes the system easier to manage, test, and extend.

5.2 Module Design

Module	Description
Input Module	Collects and validates user-provided target IP address or URL before passing it to downstream modules.
Port Scanner	Uses socket programming to iterate over a range of ports and identify those accepting connections.
Service Detector	Employs banner grabbing techniques to determine the services and versions running on open ports.
Web Vulnerability Scanner	Sends crafted HTTP requests and analyzes server responses to detect SQL injection and XSS weaknesses.
Vulnerability Analyzer	Correlates gathered data with known vulnerability patterns and classifies findings by severity.
Report Generator	Compiles all results into structured human-readable and machine-friendly output formats.

5.3 Data Flow

Data flow begins with user input (target IP or URL), which is passed to the scanning module. The scanning module collects data about open ports and running services. This data is then passed to the analysis module, which identifies vulnerabilities and assigns severity ratings. Finally, the reporting module compiles the results into a structured output for the user. This unidirectional pipeline ensures efficient processing and accurate, reproducible results.

6. SYSTEM IMPLEMENTATION

The implementation phase translates the system design into actual working code using Python. The system is developed using Visual Studio Code, and core libraries—socket for network communication and requests for HTTP interaction—provide the foundational capabilities for each module.

6.1 Port Scanning Module

The port scanning module is implemented using Python's socket library. It iterates through a defined range of ports on the target host, attempting a TCP connection on each. A successful connection indicates an open port, which is recorded for further analysis. A configurable timeout prevents the scan from hanging on filtered ports.

```
import socket
target = input("Enter target IP: ")
for port in range(1, 1025):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(0.5)
    if sock.connect_ex((target, port)) == 0:
        print(f"Port {port} is open")
    sock.close()
```

Figure 1: Port scanning implementation using Python socket library

6.2 Web Vulnerability Scanning Module

The web vulnerability scanning module uses the requests library to send crafted HTTP payloads to a target URL. For SQL injection detection, a malformed SQL payload is appended to query parameters and the response is analyzed for database error signatures. Cross-site scripting detection follows a similar pattern using script injection payloads and response content analysis.

6.3 Error Handling and Integration

The system uses exception handling to manage common errors including invalid input, network timeouts, and connection refusals. Modules are integrated through a central controller that orchestrates the scanning pipeline: input validation, port scanning, service detection, vulnerability analysis, and report generation. Proper integration ensures smooth data flow and efficient operation of the complete system.

7. RESULTS AND DISCUSSION

The vulnerability scanner was evaluated through a series of controlled experiments on local machines and sample web applications. The tests validated the accuracy and efficiency of each module, including port scanning, service detection, and web vulnerability analysis.

7.1 Port Scanning Results

The port scanning module successfully identified commonly used open ports, including Port 22 (SSH), Port 80 (HTTP), and Port 443 (HTTPS), on the test target. Scan completion time for the standard 1–1024 port range was consistently under 30 seconds on a local network, demonstrating adequate performance for practical use.

7.2 Service Detection Results

The service detection module accurately identified services running on detected open ports using banner grabbing. HTTP, HTTPS, and SSH services were correctly identified. The accuracy of detection depended on the responsiveness of the target service and the completeness of the banner information returned.

7.3 Web Vulnerability Scanning Results

The web vulnerability scanning module successfully identified SQL injection susceptibility in the test web application by detecting database error messages in the server response. XSS detection was also confirmed by observing unescaped script tags reflected in the response body. These results validate the effectiveness of the payload-based detection approach.

Module	Outcome	Notes
Port Scanning	**Successful**	*Ports 22, 80, 443 detected accurately*
Service Detection	**Successful**	*SSH, HTTP, HTTPS identified via banner grabbing*
SQL Injection	**Detected**	*Error-based detection confirmed on test app*
XSS Detection	**Detected**	*Reflected XSS identified in response body*
Report Generation	**Successful**	*Structured output with severity classification*

Table 1: Summary of experimental results across all system modules

7.4 Comparison with Existing Systems

Compared with established commercial tools, the proposed system provides a balanced, accessible solution. While advanced tools offer broader vulnerability databases and enterprise integration, the proposed scanner prioritizes usability and affordability. It is particularly well-suited for educational use, personal system assessment, and small-organization deployment where cost and simplicity are primary concerns.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

This paper presented the design and implementation of an Automated Vulnerability Scanner using Python. The system successfully integrates port scanning, service detection, web vulnerability analysis, and report generation into a single, user-friendly platform. Experimental results confirmed the system's ability to accurately detect commonly occurring vulnerabilities including open ports, exposed services, SQL injection, and cross-site scripting.

The project demonstrates that Python, combined with a modular architecture, provides an effective foundation for developing practical cybersecurity tools. The scanner addresses key limitations of existing solutions—namely high cost, complexity, and fragmentation—by delivering a free, cohesive, and accessible alternative. It is particularly valuable for educational environments, independent researchers, and small organizations seeking to improve their security posture without significant investment.

8.2 Future Enhancements

Several directions exist for future development. Integration of machine learning techniques would enable the system to detect behavioral anomalies and predict vulnerabilities beyond predefined patterns. Continuous expansion of the vulnerability signature database would ensure relevance against emerging threats. A web-based dashboard would enhance usability and provide richer data visualization.

Real-time monitoring capabilities would transform the scanner from an on-demand tool into a continuous security sentinel. Integration with cloud platforms would enable scalable assessment of cloud-native infrastructure. Incorporation of automated patch recommendations and integration with CI/CD pipelines would further embed security into the software development lifecycle.

REFERENCES

[1] **Kurose, J. F. & Ross, K. W..** "Computer Networking: A Top-Down Approach." Pearson, 7th Edition, 2017.

[2] **Stallings, W..** "Network Security Essentials: Applications and Standards." Pearson, 6th Edition, 2016.

[3] **Matthes, E..** "Python Crash Course." No Starch Press, 2nd Edition, 2019.

[4] **Seitz, J..** "Black Hat Python: Python Programming for Hackers and Pentesters." No Starch Press, 2nd Edition, 2021.

[5] **OWASP Foundation.** "OWASP Top Ten Web Application Security Risks." https://owasp.org, 2021.

[6] **Various Authors.** "Automated Vulnerability Detection Techniques in Network Security." International Journal of Computer Science and Information Security, Vol. 18, 2020.

[7] **IEEE.** "Web Application Security Testing Using Python Tools." IEEE Conference on Cybersecurity, 2022.

[8] **IRJET.** "A Study on Network Vulnerability Scanning Tools and Techniques." International Research Journal of Engineering and Technology, Vol. 9, 2022.