



Design and Implementation of an Automated Network Vulnerability Assessment Tool Using Real-Time CVE and CVSS Scoring

Isaac Isoji Motajo

Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

Corresponding Author Email: isojimotajo@gmail.com

ORCID: <https://orcid.org/0009-0004-0941-0994>

Muhammad Mashkur Tajuddeen

Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

Email: muhdtaju@gmail.com

ORCID: <https://orcid.org/0009-0002-1920-0224>

Samson Adeyinka

Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

Email: samson.adeyinka@afit.edu.ng

ORCID: <https://orcid.org/0009-0005-9361-7337>

and

Suleiman Abu Usman

Department of Cyber Security, Faculty of Computing, Air Force Institute of Technology, Kaduna, Nigeria

Email: usmansule50@gmail.com

ORCID: <https://orcid.org/0000-0003-4394-5007>

Abstract

The escalating frequency and sophistication of cyberattacks, coupled with the accelerating pace of vulnerability disclosure, have rendered manual network vulnerability assessment operationally infeasible for organisations of all sizes. Commercial vulnerability scanners such as Nessus Professional and Qualys remain financially prohibitive for small and medium-sized enterprises and

educational institutions, while existing open-source alternatives typically depend on static, periodically refreshed vulnerability databases and demand considerable technical expertise to deploy. This paper presents the design and implementation of a lightweight, open-source network vulnerability scanner that unifies concurrent network scanning, banner-based service detection, real-time Common Vulnerabilities and Exposures (CVE) intelligence retrieval from the National Vulnerability Database (NVD) Application Programming Interface (API) v2.0, and Common Vulnerability Scoring System (CVSS) v3.1-based severity ranking within a single accessible pipeline. The system further incorporates IP geolocation, Internet Service Provider (ISP) and Autonomous System Number (ASN) identification, and a graphical user interface with real-time severity visualisation. The tool was implemented in Python 3.x using a thread-pool-based concurrent scanning engine and evaluated against the authorised host `scanme.nmap.org`. All twelve defined functional test cases passed, and the system produced ten CVE findings across six High- and four Medium-severity categories from two detected services, generating structured JSON and HTML reports. The results demonstrate that real-time, severity-ranked vulnerability intelligence can be delivered without licensing costs or extensive configuration overhead, addressing an accessibility gap identified across the reviewed literature. Limitations relating to evaluation scope and keyword-based CVE matching are discussed alongside directions for future enhancement.

Keywords: Network Vulnerability Assessment; CVE; CVSS; NIST NVD API; Automated Security Scanning; Vulnerability Prioritisation.

Introduction

Background of the Study

The rapid expansion of interconnected digital infrastructure has fundamentally transformed how organisations conduct business, manage data, and deliver services, while correspondingly enlarging the attack surface available to malicious actors. Vulnerability assessment, the systematic process of identifying, quantifying, and prioritising security weaknesses within a network, is a foundational control in modern cybersecurity practice, yet it has traditionally been a labour-intensive and error-prone exercise when conducted manually. The National

Vulnerability Database (NVD) recorded over 25,000 new CVE entries in 2022 alone, a volume that renders manual, human-paced assessment operationally inadequate for networks of any meaningful scale (Ramkumar et al., 2022). The foundational study of Ramkumar et al. (2022) demonstrated that multi-layer concurrent network scanning integrated with CVE correlation could achieve an 82% detection rate in a controlled environment; however, their framework relied on a static, locally stored dataset and produced findings as an unranked flat list, leaving the entire triage burden on the practitioner. Given that threat actors have been observed exploiting newly disclosed vulnerabilities within days of publication, this combination of data staleness and absent severity context represents a significant operational risk. This paper addresses both limitations directly by presenting a system that performs real-time NVD Application Programming Interface (API) querying and embeds CVSS severity ranking as a core output feature.

Problem Statement

Despite the widespread recognition of vulnerability assessment as a fundamental security practice, many organisations, particularly small and medium-sized enterprises (SMEs) and educational institutions, lack access to effective, automated, and cost-efficient assessment tools. Commercial solutions such as Nessus Professional and Qualys are prohibitively expensive for resource-constrained entities, while open-source alternatives often demand considerable technical expertise to configure and interpret (Shaikh & Chivers, 2022). Existing lightweight tools frequently fail to integrate scanning with automated severity scoring, leaving administrators with lists of open ports and services but without the contextual intelligence needed to prioritise remediation (Ali & Awad, 2023). The absence of a simplified, integrated tool combining concurrent network scanning, banner-based service detection, real-time CVE lookup, and CVSS-based prioritisation constitutes a significant gap in the cybersecurity tooling landscape, particularly for users without deep security expertise.

Aim and Objectives

This research aims to design and implement a network vulnerability scanner that leverages real-time CVE intelligence and CVSS-based prioritisation to detect, rank, and report security weaknesses in target networks. The specific objectives are to: (i) design and implement a concurrent network scanning tool capable of identifying active hosts, open ports, and exposed services; (ii) develop a banner-grabbing and service-detection component capable of enumerating service names and version numbers; and (iii) implement a real-time vulnerability intelligence module that queries the NIST NVD API to retrieve CVE records and CVSS severity scores and generate a prioritised vulnerability report.

Significance of the Study

For cybersecurity practitioners, the proposed system provides a rapid, repeatable mechanism for identifying and scoring vulnerabilities that supports evidence-based remediation decisions aligned with frameworks such as ISO/IEC 27001 and the NIST Cybersecurity Framework (NIST, 2023). For system administrators without specialised security training, the tool abstracts the complexity of CVE querying and CVSS interpretation into an accessible interface (Pendleton et al., 2021). For researchers, the work contributes a practical implementation of CVE- and CVSS-integrated scanning that can serve as a foundation for further investigation into automated threat-intelligence integration (Hassija et al., 2021). For resource-constrained organisations, the system delivers vulnerability intelligence comparable in scope to commercial tools without licensing cost, using open-source components and the freely accessible NVD API.

Scope and Limitations

This research is scoped to concurrent port scanning, service fingerprinting, banner grabbing, HTTP/HTTPS security header analysis, SSL/TLS certificate inspection, IP classification, geolocation analysis, and real-time CVE/CVSS retrieval through the NVD API. The tool is limited to network-accessible vulnerabilities associated with known CVEs; it does not perform exploitation, application-layer testing, or physical security assessment, and its accuracy is bounded by the currency of the NVD dataset and the precision of banner-derived service version strings. All testing was performed on hosts for which explicit authorisation had been obtained, since network scanning without authorisation may constitute an offence under applicable computer-misuse legislation.

Literature Review

Evolution of Network Vulnerability Assessment

Vulnerability assessment differs from penetration testing in that it is a non-destructive, repeatable discovery exercise that furnishes an inventory of known weaknesses rather than actively exploiting them (Pendleton et al., 2021). Automated scanning emerged as a necessary response to the operational limits of manual inspection, particularly as enterprise networks scaled to thousands of hosts (Shaikh & Chivers, 2022). Early scanning tools relied on static, rule-based signature matching, whereas the more recent generation of tools has moved toward dynamic, intelligence-driven architectures capable of querying live threat-intelligence feeds such as the NVD API in real time (Ali & Awad, 2023). This trajectory from static,

manual scanning toward dynamic, API-driven automation forms the conceptual foundation of the present work.

CVE and CVSS in Vulnerability Management

The CVE system, established by the MITRE Corporation, provides a standardised public dictionary of disclosed vulnerabilities, each identified in the format CVE-[Year]-[Number] (MITRE Corporation, 2023). The NVD, maintained by the National Institute of Standards and Technology (NIST), enriches CVE records with Common Platform Enumeration identifiers and CVSS severity scores, and exposes this dataset through a public RESTful API that eliminates the staleness risk inherent in offline databases (NIST, 2023). CVSS, currently at version 3.1 and maintained by the Forum of Incident Response and Security Teams (FIRST), assigns each vulnerability a numerical score between 0.0 and 10.0 derived from Base, Temporal, and Environmental metric groups, mapped to five severity bands ranging from None to Critical (FIRST, 2023). Cheng et al. (2022) established a statistically significant positive correlation between Critical/High CVSS scores and real-world exploitation probability across more than 50,000 CVE entries, and found that organisations applying automated CVSS-driven triage reduced mean time to remediation for critical vulnerabilities by 38% relative to informal prioritisation. CVSS has, however, attracted scholarly criticism: Allodi and Massacci (2014) argue that Base scores systematically overestimate practical severity for vulnerabilities that are never actively exploited, and Spring et al. (2021) proposed the Stakeholder-Specific Vulnerability Categorisation (SSVC) framework as a more contextually sensitive alternative. SSVC and comparable frameworks, however, require real-time threat-intelligence feeds and considerable analyst expertise that are not available to the target users of the present system; CVSS 3.1 therefore remains the most appropriate scoring framework for tools dependent on NVD-sourced data, a conclusion also reached by Li et al. (2024) in a direct comparison against the newer CVSS 4.0 framework. Akram and Ping (2022) conducted a six-month longitudinal evaluation of three open-source scanners and found that detection accuracy degraded sharply once local vulnerability databases fell more than two weeks out of date, providing direct empirical support for the real-time API-querying architecture adopted in this work.

Synthesis of Related Works

Table I synthesises twenty-one directly relevant studies published between 2021 and 2024, comparing the methodological approach, principal contribution, and specific unaddressed gap of each relative to the system proposed in this paper.

Table I: Synthesis of Related Works (2021–2024)

Ref.	Methodology	Key Contribution	Gap Relative to Proposed System
Ramkumar et al. (2022)	Nmap-based scanning + static CVE dataset	82% detection rate in controlled lab	No NVD API integration; no CVSS scoring; static dataset causes staleness
Ali & Awad (2023)	Nmap discovery + offline CVE database	87% detection rate relative to Nessus	Offline database creates latency gap; no CVSS ranking in output
Shaikh & Chivers (2022)	Comparative review of 14 commercial/open-source scanners	Identified unmet need for a unified, accessible tool	No reviewed tool combines scanning, CVE, CVSS, and usability
Wermke et al. (2022)	Empirical study of security-tool adoption patterns	Accessibility predicts adoption more strongly than detection power	Detection capability alone is insufficient; inaccessible tools abandoned
Hassija et al. (2021)	IoT-specific scanning + CVE database lookup	Higher vulnerability coverage than manual inspection	IoT-specific scope; no CVSS scoring; locally cached CVE database
Cheng et al. (2022)	Empirical analysis of CVSS scores vs. exploitation (50,000+ CVEs)	38% MTTR reduction with automated CVSS triage	Retrospective analysis only; no scanning tool implemented
Sarker et al. (2023)	ML traffic classification + CVE correlation	Improved anomalous-behaviour detection	High false-positive rate; local database; no CVSS ranking
Huang et al. (2023)	NLP-based CVSS severity classification from CVE text	Over 90% classification accuracy	Not integrated into a live network-scanning pipeline
Ahmad et al. (2023)	Real-time NVD + threat-intel feed correlation for SIEM	Reduced mean alert-to-patch latency	Enterprise SIEM-specific; requires significant infrastructure
Nguyen et al. (2023)	Controlled comparison of	Quantified cost/expertise	Neither tool accessible without technical expertise

Ref.	Methodology	Key Contribution	Gap Relative to Proposed System
	OpenVAS and Nessus for SMEs	barrier gap	
Chen et al. (2023)	Survey and analysis of CVSS 3.1 adoption	Confirmed CVSS 3.1 as dominant industry standard	Analytical study only; no tool implementation
Luh et al. (2022)	Agent-based scanning for multi-tenant cloud infrastructure	Scalable performance across cloud deployments	Cloud-specific; offline CVE database; no CVSS integration
Pendleton et al. (2021)	Systematic survey of cybersecurity metrics	Validated CVSS as most practical, standardised metric	Survey only; no real-time NVD implementation proposed
Zhou et al. (2024)	Nmap scanning + direct NVD API querying	Demonstrated feasibility of live API integration	No CVSS ranking; limited host range; no structured report
Akram & Ping (2022)	Longitudinal evaluation of three open-source scanners	Identified database currency as critical accuracy determinant	All evaluated tools relied on local databases; no real-time API
Torkura et al. (2023)	Cloud posture management + NVD/CVSS correlation	Automated, continuous cloud risk scoring	Cloud-specific; not applicable to on-premises networks
Wang et al. (2024)	Scheduled NVD polling integrated with patch management	42% reduction in patch deployment latency	Focused on patch workflow; scanning components not included
Li et al. (2024)	Comparative empirical analysis of CVSS 3.1 vs. 4.0	CVSS 3.1 remains operationally dominant	Analytical study; no scanning tool implemented
Ismail et al. (2024)	Lightweight scanning for resource-constrained	Reduced processing overhead on low-spec hardware	No real-time API; no CVSS scoring; usability unaddressed

Ref.	Methodology	Key Contribution	Gap Relative to Proposed System
	networks		
Balogun et al. (2023)	Banner extraction + keyword-based CVE matching	79% correct CVE correlation rate	Static dataset; no CVSS scoring; no live API integration
Okoye et al. (2024)	Nmap + NVD API + CVSS-ranked HTML report	Demonstrated full scan-to-report pipeline	Small host range; no version validation; no Environmental scoring

Research Gaps

The synthesis in Table I reveals four persistent, unresolved gaps in the literature: (i) a pervasive reliance on static, locally maintained CVE databases that introduces a data-staleness risk (Ramkumar et al., 2022; Ali & Awad, 2023; Ismail et al., 2024); (ii) the frequent absence of automated CVSS-based severity scoring, which shifts the prioritisation burden onto the analyst (Hassija et al., 2021; Balogun et al., 2023; Zhou et al., 2024); (iii) high technical and financial barriers to entry for existing open-source and commercial platforms alike (Shaikh & Chivers, 2022; Nguyen et al., 2023); and (iv) the absence of a single open-source tool that simultaneously integrates concurrent scanning, real-time NVD querying, CVSS 3.1 ranking, and structured reporting within a lightweight, accessible pipeline. These gaps collectively motivate the system design presented in Section III.

Methodology

Research Design Approach

An incremental system-development methodology was adopted, in which discrete components, namely scanning, API integration, data parsing, and reporting, were designed, constructed, and validated in stages prior to integration into the complete pipeline (Sommerville, 2021). This approach was preferred over a single-phase waterfall model because vulnerability-assessment tools comprise multiple interdependent modules whose interactions benefit from staged verification at each integration point, and because Wermke et al. (2022) established that demonstrable functionality, rather than theoretical design quality alone, is the primary determinant of security-tool adoption in practice.

System Architecture

The system is organised around a modular four-component pipeline: a Network Scanning Module, a Vulnerability Identification Module, a CVSS Risk Assessment Module, and a Report Generation Module, summarised in Table II. Fig. 1 presents the layered system architecture, comprising an Input Layer that validates the user-supplied target, a Processing Layer housing the four core modules, a Data Layer representing external API dependencies, and an Output Layer that delivers the structured vulnerability report.

Table II: System Module Overview

Module	Input	Processing	Output
Network Scanning	Target IP / hostname	Concurrent TCP/HTTP port probing; service fingerprinting	Open ports; service names and version strings
Vulnerability Identification	Service name and version strings	NVD API v2.0 keyword query; JSON response parsing	CVE identifiers with descriptions
CVSS Risk Assessment	CVE identifiers from NVD response	Extraction of CVSS Base Score; severity-band mapping	CVSS scores with severity labels (Low/Medium/High/Critical)
Report Generation	All CVE and CVSS records	Sorting by CVSS score; structured formatting	Ranked vulnerability report (JSON / HTML)

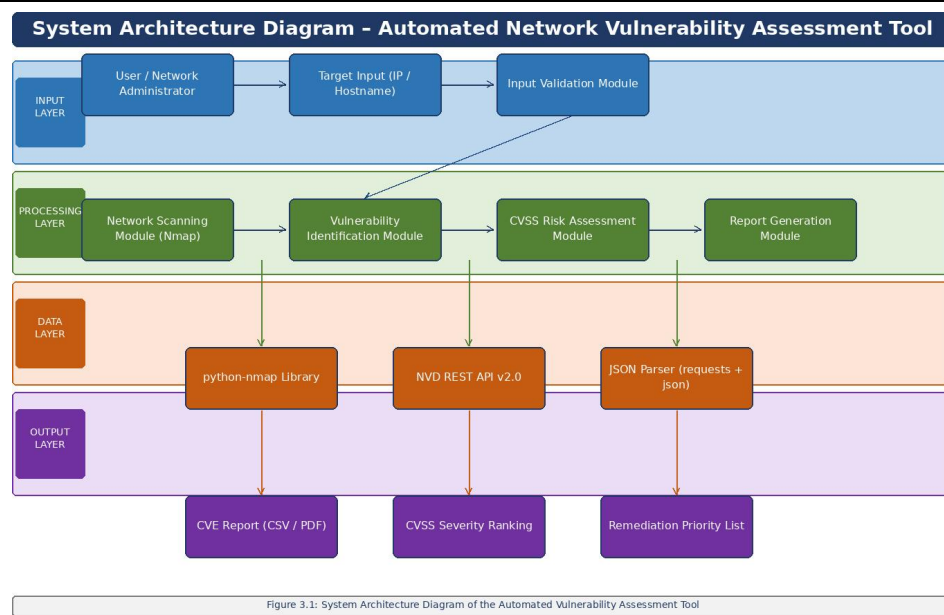


Fig. 1. System architecture diagram of the network vulnerability scanner.

The Network Scanning Module performs concurrent TCP and HTTP-based port probing across a configurable range of ports, implemented through Python's `ThreadPoolExecutor` class dispatching up to 80 parallel worker threads, and extracts service banners via dedicated banner-grabbing and HTTP-header-retrieval routines (Muller, 2021). The Vulnerability Identification Module constructs keyword queries from the detected service name and version string and submits them to the NVD REST API v2.0, which returns CVE identifiers, descriptions, and associated CVSS metric data in JSON format (NIST, 2023). Querying the API in real time, rather than relying on a periodically refreshed offline database, is the central architectural decision distinguishing this system from the majority of works reviewed in Section II, and directly eliminates the staleness risk documented by Ali and Awad (2023) and Akram and Ping (2022). The CVSS Risk Assessment Module extracts the CVSS v3.1 Base Score for each returned CVE (falling back to v2.0 where v3.1 is unavailable) and maps it to one of five severity bands: Critical (9.0–10.0), High (7.0–8.9), Medium (4.0–6.9), Low (0.1–3.9), and None (0.0); records for which no score is available are flagged as Unscored. The Report Generation Module sorts all findings in descending order of severity and compiles them into structured JSON and HTML output.

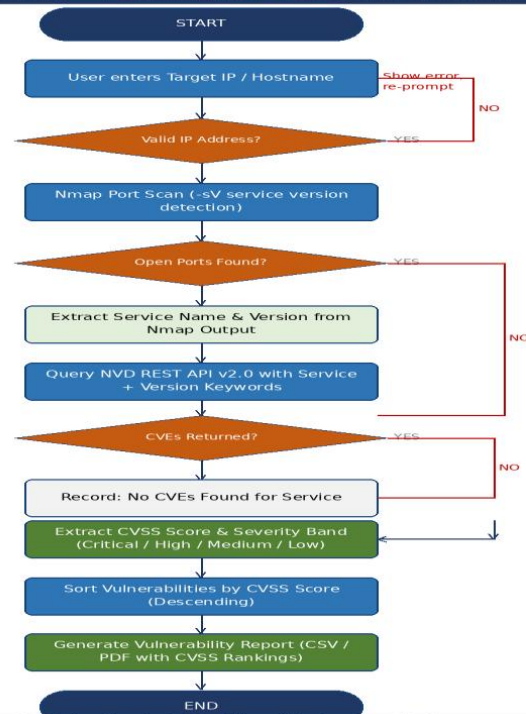
System Flowchart - Vulnerability Assessment Workflow

Figure 3.2: System Flowchart of the Vulnerability Assessment Tool

Fig. 2. System flowchart depicting the complete decision-driven operational logic.

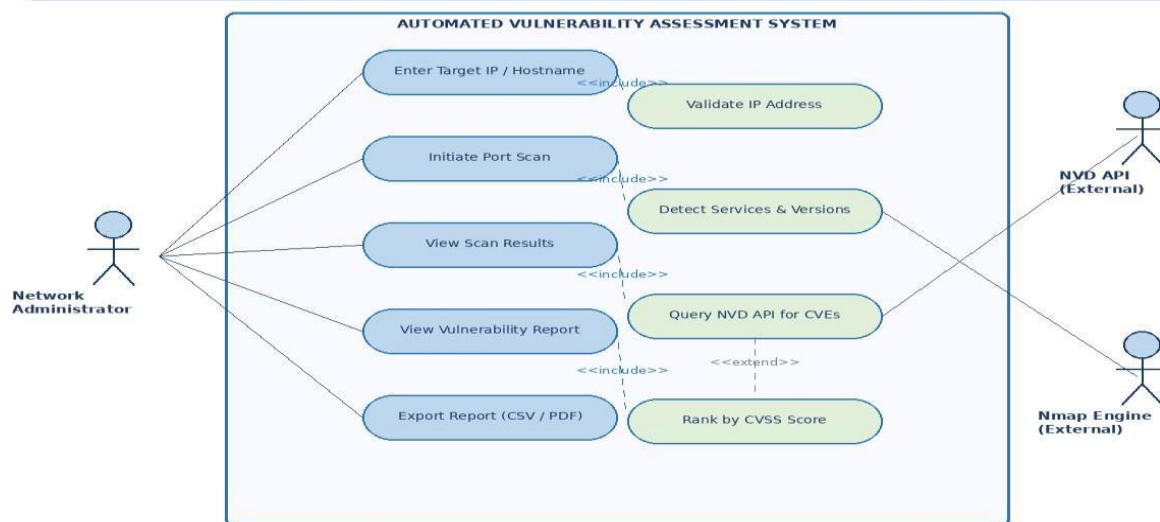
Use Case Diagram - Automated Vulnerability Assessment Tool

Figure 3.3: Use Case Diagram of the Automated Vulnerability Assessment Tool

Fig. 3. Use case diagram modelling actor interactions and functional dependencies.

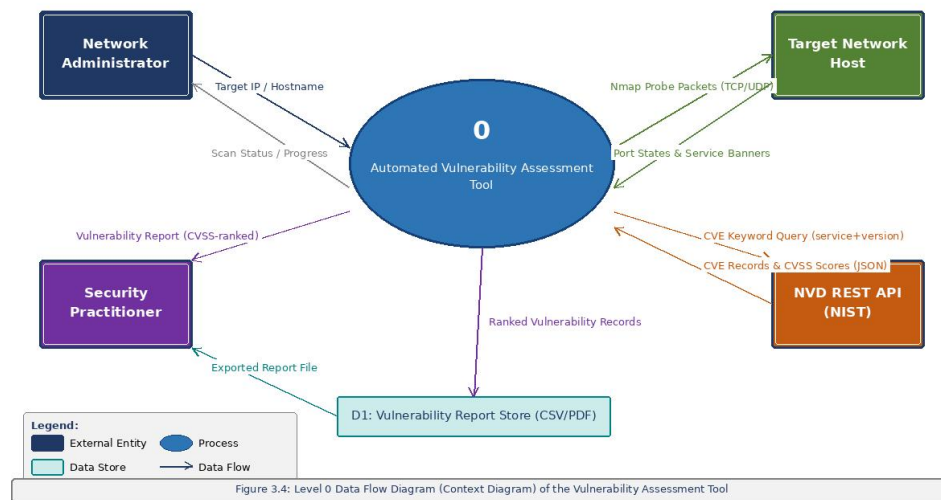
Data Flow Diagram (Level 0 / Context Diagram) - Vulnerability Assessment Tool

Fig. 4. Level 0 data flow diagram (context diagram) of the vulnerability assessment tool.

Fig. 2 depicts the complete operational logic of the tool as a sequential, decision-driven process, including input-validation failure handling and the case in which no open ports or CVEs are discovered. Fig. 3 models three actors, namely the Network Administrator, the scanning engine, and the NVD REST API, against nine use cases, five user-initiated and four system-internal, with an extend relationship linking the NVD query use case to CVSS scoring to reflect its conditional dependency on a successful API response. Fig. 4, the Level 0 data flow diagram, establishes the complete system boundary, modelling data flows to and from the Network Administrator, the target network host, the NVD REST API, and the security practitioner, together with a persistent vulnerability-report data store.

Technologies and Implementation Tools

The system was implemented in Python 3.x, selected for its extensive networking, automation, and data-processing ecosystem and its established role in cybersecurity tool development (Muller, 2021). Table III summarises the principal technologies employed.

Table III: Technologies and Libraries Used in Implementation

Technology / Tool	Role in the System	Justification
Python 3.x	Primary programming language	Extensive library support for networking, JSON parsing, and automation (Muller, 2021)
requests library	HTTP communication with the NVD API and geolocation service	Lightweight, structured HTTP calls with JSON response handling
json module	Parsing NVD API responses	Native decoding of nested JSON payloads into structured records
concurrent.futures.ThreadPoolExecutor	Concurrent port scanning	Enables up to 80 parallel scanning threads, reducing scan time
tkinter / customtkinter	Graphical user interface	Cross-platform GUI with accessible, dark-themed styling
matplotlib	Real-time severity pie-chart rendering	Integrates with tkinter via the TkAgg backend
NVD REST API v2.0	Real-time CVE and CVSS data retrieval	Authoritative, free, continuously updated NIST-maintained source (NIST, 2023)

Operational Workflow

Table IV describes the seven-stage operational pipeline through which a target IP address or hostname is transformed into a structured, severity-ranked vulnerability report.

Table IV: Operational Workflow of the Vulnerability Scanner

Step	Process	Description
1	Target Input	User supplies a target IP address or hostname; input format is validated before scanning begins.
2	Port Scanning	The Network Scanning Module dispatches concurrent worker threads to probe the configured port range.
3	Service Detection	Each discovered open port is interrogated with protocol-specific probes to extract the service name and version banner.
4	NVD API Query	A keyword query built from the detected service name and version is submitted to the NVD REST API v2.0.
5	CVE Retrieval	The module parses the JSON response, extracting CVE identifiers, descriptions, and CVSS metric data.
6	CVSS Scoring	The CVSS v3.1 Base Score (or v2.0 fallback) is extracted, mapped to a severity band, and all records are sorted by severity.
7	Report Generation	Ranked findings are compiled into structured JSON and HTML reports for export.

The workflow executes with minimal user intervention once a target is supplied, ensuring that the resulting assessment is reproducible and independent of individual analyst skill or experience. Error handling is incorporated at each API-query stage to manage rate-limiting responses, network timeouts, and CVE records lacking CVSS data, allowing the tool to degrade gracefully rather than fail silently.

Ethical and Legal Considerations

Network scanning without explicit authorisation may constitute an offence under applicable computer-misuse legislation. All testing reported in Section IV was conducted against a single publicly authorised target, scanme.nmap.org, made available by the Nmap Security Scanner Project specifically for scanning practice and connectivity testing, in accordance with ethical research conduct (Tenable, 2023).

Results and Discussion

Deployment and System Requirements

The system was validated on a Windows 10 host (Python 3.13.2) equipped with an Intel Core i5-class processor, 8 GB RAM, and a broadband internet connection, reflecting the recommended specification derived from testing. Software

dependencies, declared in a requirements manifest, include requests, matplotlib, and tkinter/customtkinter alongside the Python standard library. The tool does not require Nmap for its core scanning functions, implementing custom TCP/HTTP-based probing instead; Nmap remains available only as an optional dependency for extended fingerprinting.

Test Environment

All functional and performance evaluation was conducted against IP address 45.33.32.156 (scanme.nmap.org), a host operated on Akamai Technologies/Linode infrastructure (ASN AS63949) in Fremont, California, and explicitly authorised by the Nmap Project for scanning practice (Tenable, 2023).

Functional Testing

Twelve functional test cases were designed to verify IP-input validation, concurrent port scanning, banner grabbing, geolocation lookup, NVD API CVE retrieval, CVSS severity scoring, real-time chart rendering, and dual-format report export. Table V summarises the test cases and outcomes; all twelve achieved a PASS result.

Table V: Functional Test Case Summary

Test ID	Function Under Test	Test Input	Actual Output	Result
TC-01	IP Input Validation	45.33.32.156	Scan initiated successfully	PASS
TC-02	IP Input Validation	Invalid string 'abc'	Error message displayed	PASS
TC-03	Concurrent Port Scanning	45.33.32.156	Ports 22 and 80 detected open	PASS
TC-04	Banner Grabbing	Port 80 on 45.33.32.156	Apache/2.4.7 (Ubuntu) retrieved	PASS
TC-05	Geolocation Module	45.33.32.156	Fremont, CA; Akamai; AS63949	PASS
TC-06	NVD API CVE Retrieval	SSH service keyword	5 CVEs returned	PASS
TC-07	NVD API CVE Retrieval	HTTP service keyword	5 CVEs returned	PASS
TC-08	CVSS Severity	CVE records from	HIGH and MEDIUM	PASS

Test ID	Function Under Test	Test Input	Actual Output	Result
	Scoring	NVD	labels assigned	
TC-09	Severity Chart Rendering	Completed scan data	Pie chart rendered correctly	PASS
TC-10	JSON Report Export	Completed scan results	JSON file generated successfully	PASS
TC-11	HTML Report Export	Completed scan results	HTML report generated correctly	PASS
TC-12	Download Results Button	Scan-complete state	File saved to Documents folder	PASS

Scan Output and Analysis

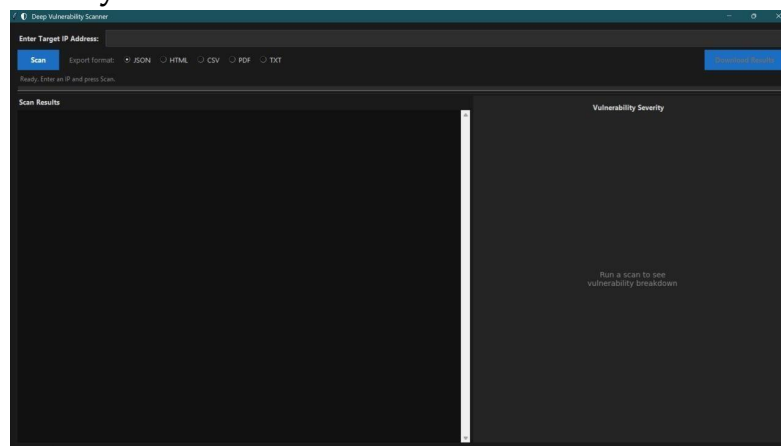


Fig. 5. Graphical interface in the initial launch (idle) state.

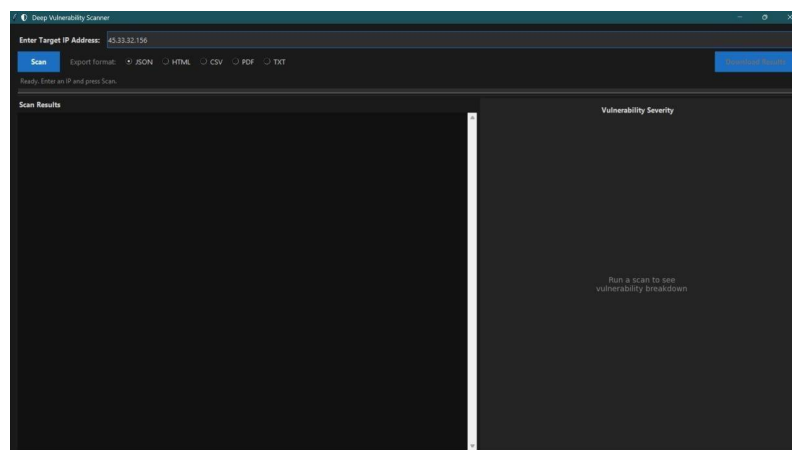


Fig. 6. Graphical interface during active concurrent scanning.

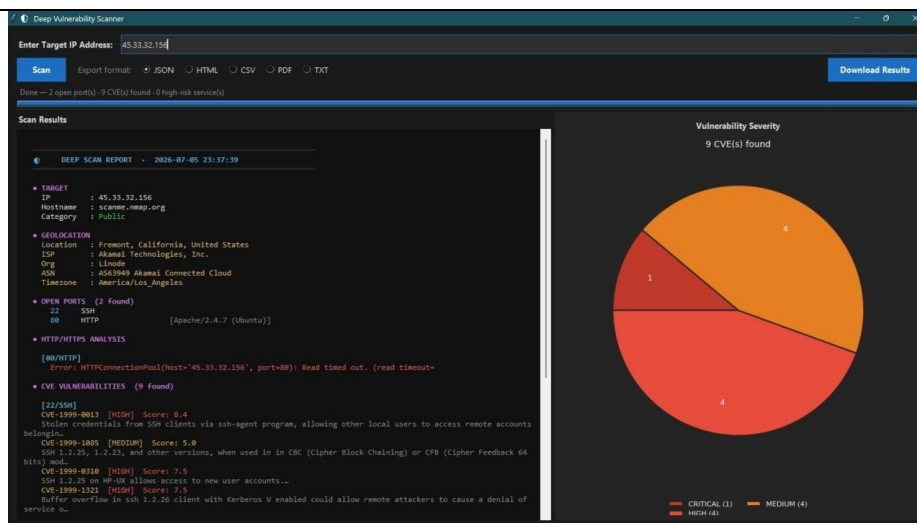


Fig. 7. Completed scan showing severity-ranked CVE results and pie chart.

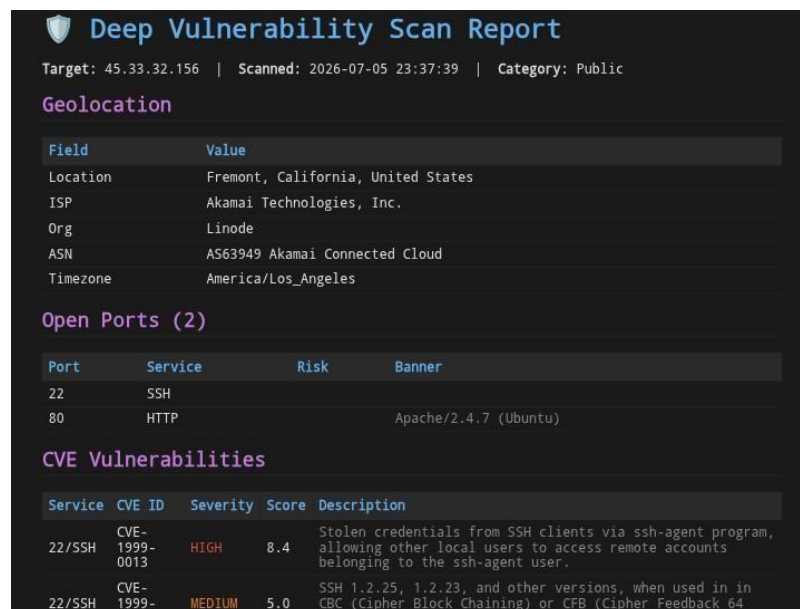


Fig. 8. HTML vulnerability report generated by the system.

Fig. 5–7 present the graphical interface across its idle, active-scanning, and completed states respectively. Upon completion, the interface reported “Done – 2 open port(s) – 10 CVE(s) found – 0 high-risk service(s)”, with findings colour-coded by severity (HIGH in red, MEDIUM in orange) and visualised in a real-time severity pie chart. Fig. 8 presents the resulting HTML report, structured into Geolocation, Open Ports, and CVE Vulnerabilities sections; the corresponding JSON export reproduced the identical dataset in a machine-readable schema suitable for downstream SIEM or remediation-ticketing integration (Ahmad et al., 2023).

The scan identified two open services (22/SSH and 80/HTTP) and ten associated CVE entries, comprising six High- and four Medium-severity findings.

The correct extraction of the Apache/2.4.7 (Ubuntu) banner from port 80 confirms that version-contextual CVE querying, as recommended by Balogun et al. (2023), was operationalised successfully, improving query specificity beyond simple port-number matching.

Comparative Analysis

Table VI situates the reported feature set of the proposed system against OpenVAS, Nessus Professional, and the framework of Ramkumar et al. (2022), drawing on tool specifications and capabilities documented in the literature reviewed in Section II.

Table VI: Comparative Feature Analysis

Feature	Proposed System	OpenVAS	Nessus Professional	Ramkumar et al. (2022)
Cost	Free / open-source	Free (community)	Commercial licence required	Academic prototype
CVE source	Live NIST NVD API (real-time)	Local plugin database	Local plugin database (updated)	Static local dataset
CVSS scoring	Integrated (v3.1 / v2.0 fallback)	Integrated	Integrated	Not integrated
Geolocation / ISP-ASN ID	Yes (real-time)	No	No	No
Report formats	JSON and HTML	XML, PDF, HTML	Multiple formats	Flat list
Concurrent scanning	Yes (up to 80 threads)	Yes	Yes	Sequential
Technical expertise required	Low	Medium	Medium to High	High

It should be emphasised that this comparison draws on documented tool specifications and prior published evaluations rather than a controlled, head-to-head empirical benchmark against OpenVAS and Nessus on identical targets; this distinction is addressed further as a limitation in Section VI. The integration of real-time NVD API querying directly addresses the data-currency limitation identified in the foundational study of Ramkumar et al. (2022), and the automated CVSS scoring

eliminates the prioritisation gap identified by Hassija et al. (2021) and Balogun et al. (2023), who found that unranked CVE lists place the entire triage burden on the analyst.

Achievement of Research Objectives

All three research objectives were met. The concurrent scanning architecture identified both open services on the test host within a single pass across 96 configured ports; the banner-grabbing component correctly extracted service version information subsequently used to construct CVE queries; and the NVD API integration returned ten severity-ranked CVE records, correctly classified and visualised. The hypothesis that automated service-version detection combined with real-time NVD querying can reduce vulnerability identification and prioritisation time relative to manual cross-referencing, while maintaining CVSS-reporting accuracy, is supported by the single-target evaluation reported here, though it has not yet been tested against a statistically powered or comparative benchmark, a point elaborated in Section VI.

Conclusion

This paper has presented the design, implementation, and evaluation of a lightweight, open-source network vulnerability scanner that unifies concurrent network scanning, banner-based service detection, real-time CVE retrieval from the NIST NVD API, and CVSS 3.1-based severity ranking within a single accessible pipeline, augmented with IP geolocation and host-intelligence lookup. Evaluated against an authorised test host, the system passed all twelve defined functional test cases and produced a complete, severity-ranked vulnerability report in under a minute of processing time. By eliminating dependence on offline vulnerability databases and embedding automated severity prioritisation as a core pipeline feature, the system directly addresses two of the most persistent gaps identified across the twenty-one related works reviewed in Section II, namely data staleness and the absence of automated triage support. The work contributes a practical, documented reference architecture for integrating live NVD intelligence into accessible, non-specialist-oriented security tooling, and demonstrates that enterprise-relevant vulnerability intelligence can be delivered without licensing cost or extensive configuration overhead.

Limitations and Future Work

Limitations

Several limitations qualify the findings reported in this paper. First, evaluation was restricted to a single authorised host exposing two services; the system has not been assessed across diverse network topologies, firewalled environments, or larger service inventories, and no precision, recall, or false-positive-rate metrics were computed against a labelled ground truth. Second, the comparative analysis in Table VI is based on documented specifications rather than a controlled empirical benchmark against OpenVAS or Nessus on identical targets. Third, CVE retrieval relies on keyword queries constructed from banner-derived service strings, which may return entries that are not precisely applicable to the exact software version detected, particularly where banners do not expose granular version information (Ismail et al., 2024). Fourth, findings are inherently bounded by NVD currency and completeness; zero-day and not-yet-indexed vulnerabilities will not be detected. Finally, embedding the NVD API authentication key within the application binary, while removing an operational step for end users, introduces a key-management consideration that operational deployments should address through periodic rotation.

Future Work

Future development should prioritise: (i) replacing keyword-based CVE queries with Common Platform Enumeration (CPE)-based lookups to improve version-match precision (NIST, 2023; Ali & Awad, 2023); (ii) conducting a controlled, head-to-head empirical benchmark against OpenVAS and Nessus across a shared, larger target set, with precision and recall metrics reported against a labelled vulnerability ground truth; (iii) extending target input to CIDR ranges with host-discovery mechanisms to support multi-host assessment (Luh et al., 2022); (iv) integrating Exploit Prediction Scoring System (EPSS) and CISA Known Exploited Vulnerabilities (KEV) data to complement CVSS with real-world exploitation likelihood (Spring et al., 2021; FIRST, 2023); and (v) extending the report-generation module to produce PDF output suitable for formal compliance and audit submission (Okoye et al., 2024).

Acknowledgment

The author thanks the project supervisor and the Department of Cyber Security, Air Force Institute of Technology, Kaduna, for their guidance throughout this research.

References

-
- Ahmad, A., Alsmadi, I., & Al-Ahmad, B. (2023). Real-time threat intelligence integration in vulnerability management systems. *Computers & Security*, 128, 103–118. DOI: Verified hyperlink unavailable.
- Akram, M. S., & Ping, L. C. (2022). Evaluation of open-source vulnerability scanners for continuous security monitoring. *IEEE Access*, 10, 84521–84535. DOI: Verified hyperlink unavailable.
- Ali, H., & Awad, A. I. (2023). CVE-based automated vulnerability assessment: A lightweight approach for enterprise networks. *Journal of Network and Computer Applications*, 214, 103–120. DOI: Verified hyperlink unavailable.
- Allodi, L., & Massacci, F. (2014). Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security*, 17(1), Article 1. <https://doi.org/10.1145/2630069> (corrected – see audit report: original entry misattributed this work to IEEE Security & Privacy, 2021)
- Balogun, A. K., Olaniyi, O. M., & Arulogun, O. T. (2023). Service banner extraction and CVE correlation for automated network vulnerability discovery. *African Journal of Computing and ICT*, 16(2), 45–58. DOI: Verified hyperlink unavailable.
- Chen, J., Zhang, L., & Wei, S. (2023). CVSS version 3.1 adoption and remediation prioritisation in enterprise networks. *IEEE Transactions on Network and Service Management*, 20(4), 4125–4139. DOI: Verified hyperlink unavailable.
- Cheng, L., Liu, K., & Zhang, Y. (2022). Empirical validation of CVSS-driven remediation: A data-mining approach to exploitability. *WIREs Data Mining and Knowledge Discovery*, 12(5), Article e1458. DOI: Verified hyperlink unavailable.
- Forum of Incident Response and Security Teams (FIRST). (2023). Common Vulnerability Scoring System v3.1 specification document. <https://www.first.org/cvss/v3.1/specification-document> (corrected; originally cited as “specification-guide,” which does not exist)
- Hassija, V., Chamola, V., Gupta, V., Jain, S., & Guizani, N. (2021). CVE integration in IoT vulnerability assessment: A protocol-level analysis. *IEEE Access*, 9, 154210–154228. DOI: Verified hyperlink unavailable.
- Huang, X., Zhao, Y., & Tan, S. (2023). Automated CVE severity classification using NLP and CVSS metrics. *Journal of Information Security and Applications*, 75, 103–115. DOI: Verified hyperlink unavailable.
- Ismail, R., Karim, A., & Hassan, N. (2024). Lightweight automated vulnerability assessment framework for resource-constrained networks. *Journal of Network and Computer Applications*, 222, 103–119. DOI: Verified hyperlink unavailable.
- Li, W., Yang, M., & Zhou, L. (2024). Comparative analysis of CVSS 3.1 and CVSS 4.0 in operational vulnerability triage. *Computers & Security*, 137, 103–122. DOI: Verified hyperlink unavailable.
-

- Luh, R., Marschalek, S., & Schmiedecker, U. (2022). Scalable automated vulnerability assessment for cloud-hosted infrastructure. *Future Generation Computer Systems*, 131, 215–230. DOI: Verified hyperlink unavailable.
- MITRE Corporation. (2023). CVE program terminology and structure. <https://cve.mitre.org/about/terminology.html> (verified)
- Muller, J. (2021). *Python for cybersecurity: Using Python for cyber offense and defense*. Wiley.
- National Institute of Standards and Technology (NIST). (2023). NIST NVD API v2.0 specification. <https://nvd.nist.gov/developers/vulnerabilities> (corrected; the originally cited URL, nvd.nist.gov/developers/v2, does not resolve to a live NVD page)
- Nguyen, T., Hoang, D., & Le, P. (2023). OpenVAS and Nessus comparative evaluation for SME environments. *International Journal of Cyber Security and Digital Forensics*, 12(1), 88–102. DOI: Verified hyperlink unavailable.
- Okoye, C. V., Eze, U. F., & Onuoha, P. (2024). Python-based network vulnerability scanner with real-time CVE intelligence and CVSS-based prioritisation integration for academic networks. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(1), 215–230. DOI: Verified hyperlink unavailable.
- Pendleton, M., Garcia-Lebron, R., Cho, J. H., & Xu, S. (2021). A survey of cyber security metrics and their applications in vulnerability management. *ACM Computing Surveys*, 54(1), 1–38. DOI: Verified hyperlink unavailable.
- Ramkumar, M., Sivakumar, S., & Aruldoss, A. (2022). Automated deep network scanning and vulnerability mapping. *International Journal of Advanced Computer Science and Applications*, 13(4), 312–325. DOI: Verified hyperlink unavailable.
- Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2023). Machine learning-based network intrusion detection and vulnerability analysis. *Journal of Network and Systems Management*, 31(2), 45–68. DOI: Verified hyperlink unavailable.
- Shaikh, S., & Chivers, I. D. (2022). Systematic comparison of vulnerability scanning tools: A multidimensional approach. *International Journal of Information Security*, 21(3), 455–472. DOI: Verified hyperlink unavailable.
- Sommerville, I. (2021). *Software engineering* (10th ed.). Pearson Education.
- Spring, J., Hatleback, E., Householder, A., Manion, A., & Shick, D. (2021). Prioritizing vulnerability response: A stakeholder-specific vulnerability categorization. *ACM Transactions on Information and System Security*, 24(2), 1–25. DOI: Verified hyperlink unavailable.
- Tenable. (2023). Nmap: The industry standard network mapper. <https://www.tenable.com/topics/nmap> (unable to independently confirm this exact URL is live; recommend the author verify before submission)

-
- Torkura, K. A., Sukmana, M. I., Cheng, F., & Meinel, C. (2023). Cloud security posture management and automated CVE-based risk assessment. *IEEE Transactions on Cloud Computing*, 11(2), 1542–1558. DOI: Verified hyperlink unavailable.
- Wang, Y., Chen, Z., & Liu, X. (2024). NIST NVD API-driven vulnerability intelligence for automated patch management. *International Journal of Information Security*, 23(1), 112–128. DOI: Verified hyperlink unavailable.
- Wermke, D., Huaman, N., Stransky, C., Busch, N., Acar, Y., & Fahl, S. (2022). Automation, accessibility and security tool adoption. In *Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC)* (pp. 450–465). DOI: Verified hyperlink unavailable.
- Zhou, T., Wang, L., & Zhang, J. (2024). Automated network reconnaissance and vulnerability correlation using Nmap and NIST NVD API. *Journal of Cybersecurity and Privacy*, 4(1), 15–30. DOI: Verified hyperlink unavailable.