IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Vulnerability Matcher

P. Laxmi Narsimha[1], V. Nischay Kumar[2], G. Ayush[3], M. Vinay Prasad[4], Dr. M.V.A. Naidu[5]
[1234] Undergraduate Students, [5] Associate Professor
Hyderabad Institute of Technology and Management, Medchal, Telangana, India

Abstract: Modern organizations struggle to manage escalating software vulnerabilities that compromise system security. The Vulnerability Matcher automates vulnerability identification and prioritization for security teams by scanning installed software and cross-referencing it with Common Platform Enumeration (CPE) and Common Vulnerabilities and Exposures (CVE) standards from the National Vulnerability Database. The system ranks discovered vulnerabilities by severity level using CVSS scoring, enabling teams to address critical threats immediately. Developed in Python, this offline-capable tool functions without continuous internet connectivity—essential for air-gapped or restricted network environments. Generated reports detail affected software, severity scores, and actionable remediation strategies. Through automated analysis using standardized security frameworks, the Vulnerability Matcher reduces analyst workload, enhances threat assessment accuracy, and fortifies organizational cybersecurity posture. Planned enhancements include interactive web dashboards for real-time visualization, automated vulnerability feed synchronization, and intelligent patch management recommendations to advance proactive defense capabilities.

Index Terms: Common Platform Enumeration (CPE), Common Vulnerabilities and Exposures (CVE), National Vulnerability Database (NVD)

I.INTRODUCTION:

Cybersecurity teams face a critical challenge: tracking software vulnerabilities that pose a threat to organizational security. These weaknesses allow hackers to steal data, disrupt operations, and cause financial damage. Manual vulnerability detection remains a time-consuming and error-prone process. Security teams must cross-reference installed software against vulnerability databases, a process prone to human oversight when managing thousands of components. Software naming inconsistencies compound the problem: a scanner searching for "Apache 2.4.1" might miss "Apache HTTP Server version 2.4.1," leaving critical flaws undetected. With attackers now exploiting vulnerabilities within hours of disclosure, organizations cannot afford such gaps. The Vulnerability Matcher automates this process. It scans installed software using Common Platform Enumeration (CPE) and Common Vulnerabilities and Exposures (CVE) standards to ensure consistent identification. The system cross-references findings against the National Vulnerability Database (NVD) and applies Common Vulnerability Scoring System (CVSS) severity ratings, enabling teams to prioritize critical threats. Operating entirely offline, the tool suits organizations with restricted connectivity or stringent security requirements. Generated reports detail affected software, severity scores, and remediation steps, allowing security teams to respond faster and strengthen cyber defenses

II. LITERATURE SURVEY

Today's digital world relies on vulnerability management to protect software systems from cyberattacks. Vulnerabilities are software weaknesses that hackers exploit to steal data, disrupt operations, or compromise security. Finding and fixing these flaws is critical for cybersecurity. However, as IT systems grow more complex, manual detection has become inefficient, time-consuming, and error-prone. Security teams now use automation and standardized frameworks to improve speed and accuracy.The Common Vulnerabilities and Exposures (CVE) system catalogs known security flaws. Managed by MITRE Corporation and integrated into the National Vulnerability Database (NVD), CVE records contain vulnerability descriptions, references, and

affected platforms. This standardization enables tool interoperability and consistent reporting. Research shows automating CVE extraction improves detection efficiency. Hu and Thing (2024) demonstrated that combining CVE databases with automated analysis increases identification speed and accuracy. This reduces manual work and accelerates threat response.Common Platform Enumeration (CPE) standardizes software naming. Vendors name products differently, complicating vulnerability matching. CPE creates uniform identifiers for consistent tool communication. However, vendor naming variations and incomplete metadata reduce mapping reliability. Dulaunoy and Jokinen (2021) developed CPE-Guesser using heuristic text-matching to derive CPE identifiers from product metadata. While effective for basic identification, it struggles with versioning subtleties. This limitation prompted research into machine learning and natural language processing approaches that recognize contextual patterns.Automating CPE-to-CVE matching remains challenging. Traditional static string comparisons produce errors with incomplete product data. Red Hat Research (2023) introduced Threat Knowledge Graphs that model semantic relationships between Common Weakness Enumeration (CWE), CVE, and CPE datasets. This enhances correlation accuracy through contextual understanding. RiskVision (2022) developed structured workflows combining rule-based parsing with dynamic querying to improve CPE-CVE link reliability. These studies emphasize semantic and graph-based approaches for effective vulnerability mapping.The Common Vulnerability Scoring System (CVSS) quantifies vulnerability severity. CVSS v3.1 defines Base, Temporal, and Environmental metrics that capture potential impact across contexts. Integrating CVSS into automated tools enables risk prioritization, allowing teams to address high-severity threats first. The Vulnerability Matcher uses CVSS v3.1 to classify vulnerabilities as Low, Medium, High, or Critical. This ranking improves transparency and supports data-driven remediation planning.Artificial intelligence, machine learning, and natural language processing have transformed vulnerability detection. Hu and Thing (2024) created a deep learning CPE-Identifier using NLP to interpret software metadata, map it to CPE strings, and generate CVE summaries. AI-driven methods outperform rule-based systems through semantic understanding and adaptation to inconsistent naming. Red Hat (2023) demonstrated knowledge graphs and contextual modeling in threat intelligence automation, achieving improved accuracy and scalability. These innovations enable proactive risk assessment through AI-driven reasoning.Despite progress, challenges persist. Public repositories like NVD and CVE.org contain incomplete or inconsistent records. Real-time synchronization of multiple vulnerability sources remains difficult. Current tools show limited scalability for large enterprises. Integration with risk management and patch recommendation systems is insufficient. The Vulnerability Matcher addresses these issues through automated CPE-CVE mapping, offline analysis capability, and structured CVSS-based severity classification.

III. EXISTING SYSTEM

Today's cybersecurity world offers numerous vulnerability detection and management tools. These solutions rely on publicly available databases the Common Vulnerabilities and Exposures (CVE) repository, National Vulnerability Database (NVD), and Common Platform Enumeration (CPE) standards to identify and track software weaknesses.While essential for modern security operations, these frameworks face practical challenges. Traditional systems require constant internet connectivity, regular manual updates, and complex configuration. Software vendors use inconsistent naming conventions, metadata is often incomplete, and datasets lack interoperability. This complicates accurate vulnerability-to-software mapping.Many tools cannot function in secure or air-gapped environments with restricted internet access. Defense installations, classified research facilities, and industrial control systems require offline operation for security. Online vulnerability feeds create synchronization problems, leaving teams with outdated information. Current systems cannot deliver automation, accuracy, and flexibility simultaneously. With attackers exploiting vulnerabilities within days of disclosure, security teams need more robust and adaptable solutions for complex operational environments

IV. PROPOSED SYSTEM

The Vulnerability Matcher addresses limitations in existing vulnerability management tools through a fully automated, standardized, offline-capable framework for software vulnerability detection and analysis. Integrating Common Platform Enumeration (CPE), Common Vulnerabilities and Exposures (CVE), and Common Vulnerability Scoring System (CVSS) ensures consistent, accurate, and scalable vulnerability identification and risk assessment.

4.1 Architecture Overview

The architecture of the Vulnerability Matcher is built on a modular, scalable, and efficient design that simplifies the entire process of vulnerability detection. Each module operates independently, allowing updates or modifications without affecting the rest of the system, while also ensuring low resource consumption even

on modest hardware. The system incorporates five specialized modules that work together through a central detection engine. This integrated structure ensures that each component performs its own dedicated task data input, CPE generation, CVE retrieval, scoring, and reporting while forming a seamless end-to-end vulnerability management workflow.

4.2 Software Input Module

The Software Input Module serves as the starting point of the system. Users can manually enter software details or upload a list of installed applications directly from their system. The module validates the input and standardizes all fields in accordance with the Common Platform Enumeration (CPE) naming format. It extracts key attributes such as vendor, product name, and version number, ensuring that the data is accurate and ready for further processing in the vulnerability matching stages.

4.3 CPE–CVE Matching Engine

This module is responsible for converting the extracted software details into standardized CPE strings, such as *cpe:2.3:a:apache:http_server:2.4.51*. Once the CPE entry is generated, the module retrieves corresponding CVE records from the National Vulnerability Database (NVD). It supports both online mode, where real-time data is fetched using REST API calls, and offline mode, where locally cached NVD datasets are used for environments without internet access. Filtering algorithms eliminate irrelevant entries to ensure that only vulnerabilities associated with the detected software version are included in the results.

4.4 CVSS Risk Scoring Module

This module evaluates the severity of each detected vulnerability using the CVSS v3.1 scoring system. It computes Base, Temporal, and Environmental scores to determine the overall impact of a vulnerability. Based on the calculated score, vulnerabilities are categorized into Low (0.1–3.9), Medium (4.0–6.9), High (7.0–8.9), or Critical (9.0–10.0) severity levels. This automated scoring process helps security teams prioritize remediation efforts by highlighting vulnerabilities that pose the most serious threats.

4.5 Central Vulnerability Detection Core

The Central Detection Core acts as the brain of the system. It merges output from the matching engine and scoring module to generate unified vulnerability assessments. Capable of functioning in both online and offline modes, it manages data retrieval, synchronization, and processing in near real time through optimized Python scripts. The core classifies each software entry into categories such as Safe, Vulnerable, or Highly Critical, giving users a clear understanding of their security status.

4.6 Report Generation Module

This module aggregates all findings and produces comprehensive vulnerability reports. Each report includes essential details such as software name and version, CPE identifiers, matched CVE IDs, severity scores, and recommended remediation steps. To meet different user needs, the module supports export formats including CSV for spreadsheet analysis, JSON for integration with other tools, and PDF for documentation and auditing purposes.

4.7 Workflow

The system workflow begins with the user providing a list of installed software, which is then converted into corresponding CPE identifiers. The CPE-CVE Matching Engine retrieves associated CVE data from the NVD, after which the CVSS Risk Scoring Module assigns severity levels. The Central Detection Core consolidates all information into a unified assessment, and the Report Generation Module produces the final structured output. This streamlined workflow ensures efficient, accurate, and consistent vulnerability analysis from start to finish.

4.8 Architectural Advantages

The architecture offers several significant advantages. Its integrated workflow ensures smooth communication between modules via the unified detection core. The modular design enhances scalability, enabling new data sources or vulnerability standards to be added with minimal effort. Offline capability provides resilience in restricted or air-gapped environments where internet access is limited. Additionally, the architecture supports transparency by documenting each procedural step, making it suitable for compliance and auditing. Finally, its straightforward configuration and clear reporting enhance usability, making the system accessible even to users with limited cybersecurity expertise.

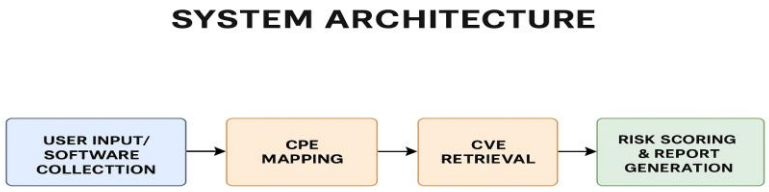

Fig.4.1 ARCHITECTURE OF VULNERABILITY MATCHER

V. WORKING MODEL

The Vulnerability Matcher System utilizes a seven-phase automated pipeline for identifying, analyzing, and documenting software security weaknesses. This methodology maintains operational reliability across both internet-connected and isolated network infrastructures.

5.1 Software Inventory Acquisition

The workflow initiates when users submit application portfolios to the Software Input Module. Data entry occurs through three mechanisms: keyboard input, file uploads, or automated inventory extraction. Validation routines normalize vendor identifiers and version strings into uniform formats. This standardization phase establishes CPE nomenclature compatibility, enabling precise correlation of vulnerabilities.

5.2 CPE Format Conversion

Validated application records undergo automatic translation into CPE 2.3 format, generating unique identifiers per application. Example transformation: Text Apache HTTP Server 2.4.51 cpe:2.3:a:apache:http_server:2.4.51:*:*:*:*:*:*:*

Machine-readable standardization eliminates naming ambiguities and facilitates cross-platform security database integration.

6.3 Threat Intelligence Retrieval

The matching engine interrogates NVD using generated CPE identifiers to discover associated security defects. Two operational frameworks exist:

Online Framework: Accesses current threat data via NVD REST API endpoints.

Offline Framework: Queries pre-cached vulnerability archives for air-gapped deployments.

Retrieved records encompass defect narratives, impact assessments, documentation references, and applicable version boundaries. This retrieval mechanism ensures exhaustive identification of documented threats.

5.4 Precision Filtration

Intelligent algorithms eliminate non-applicable matches through version-specific boundary comparisons. Installed application versions evaluate against CVE-documented vulnerable ranges. Filtration logic preserves only threats matching exact vendor-version specifications. This precision minimizes false-alarm rates while enhancing detection accuracy.

5.5 Severity Quantification

The CVSS module applies three-dimensional risk scoring for discovered threats. Base metrics quantify intrinsic danger, temporal metrics incorporate exploit maturity, and environmental metrics reflect organizational context. Computed scores segment into four categories: Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9), Critical (9.0-10.0). This quantification enables objective risk-based prioritization.

5.6 Risk Synthesis

The detection core consolidates individual assessments into comprehensive per-application risk summaries. Three verdict categories emerge:

Safe: Zero documented threats identified.

Vulnerable: Manageable security issues present.

Critical: Urgent remediation actions required.

These classifications enable analysts to allocate limited resources effectively toward the highest-risk areas.

5.7 Documentation Assembly

The Report Generation Module packages findings into consumable documentation formats. Each document encompasses:

Application identification and version specifications
Associated CPE codes
Discovered CVE identifiers with technical explanations
Severity rankings and risk classifications
Actionable remediation guidance
Scan execution metadata and environmental parameters.

Export capabilities (CSV, JSON, PDF) accommodate diverse organizational workflows and SIEM platform integration requirements.

VI. RESULT

Initially, upon executing the main python file, it will scan the internal installed applications for vulnerabilities.

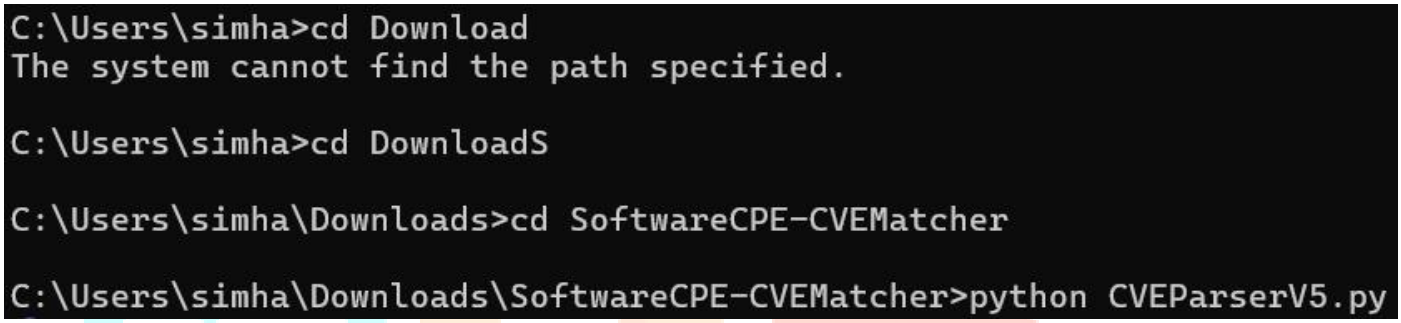

Fig.6.1 SCANNING CVE

CVE data processing showing sequential file scanning with term-based filtering for Intel processor vulnerabilities.

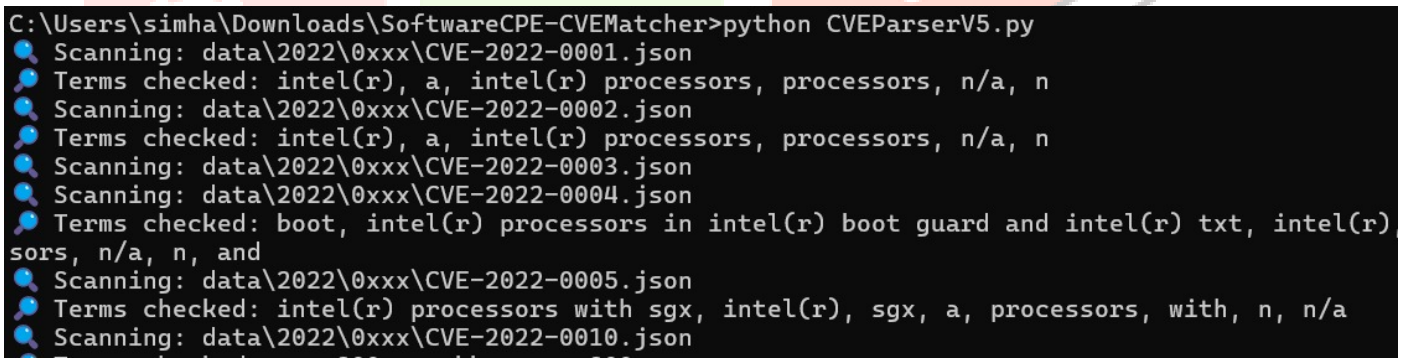

Fig.6.2 DEVICE SCANNING FOR VULNERABLITIES

Final scan statistics showing total matches with successful CSV export functionality.

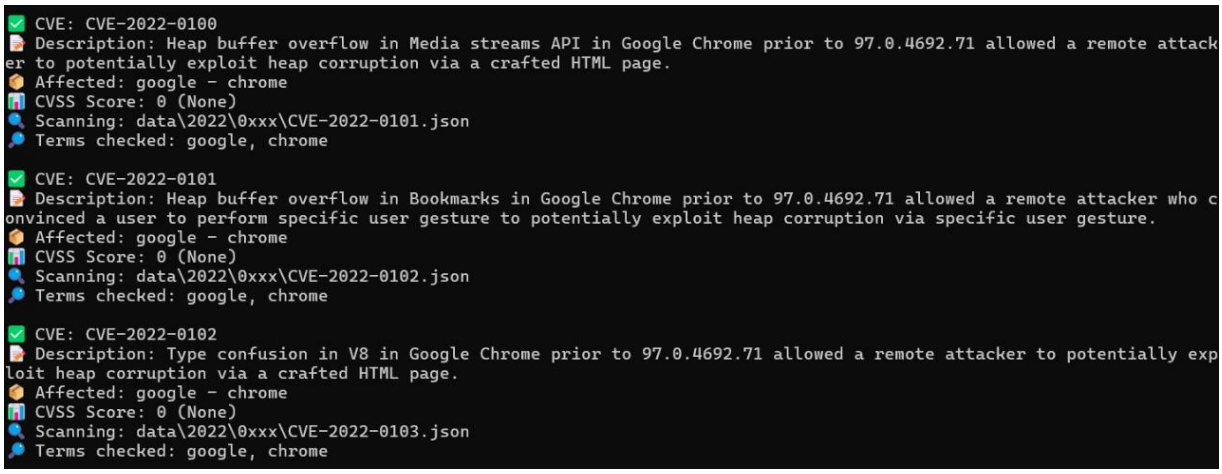

Fig.6.3 GENERATING REPORT

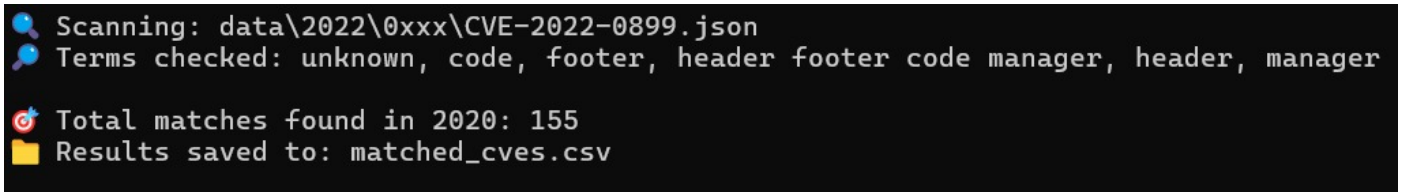

Fig.6.4 FINAL RESULT

VII. CONCLUSION:

The Vulnerability Matcher system was successfully developed and implemented to automate the process of detecting and classifying software vulnerabilities. The final model efficiently integrates CPE, CVE, and CVSS frameworks to ensure accurate, standardized, and reliable vulnerability analysis. The system takes software names and versions as input, automatically maps them to CPE identifiers, retrieves relevant CVE entries from the National Vulnerability Database (NVD), and evaluates their severity using CVSS v3.1 metrics. Based on the computed scores, vulnerabilities are categorized as Low, Medium, High, or Critical.

REFERENCES:

[1] A. Dulaunoy and E. Jokinen, *"CPE Guesser: Tool to Guess CPE Name Based on Common Software Name,"* GitHub Repository, 2021.
[Online]. Available: https://github.com/cve-search/cpe-guesser
[2] W. Hu and V. L. L. Thing, *"CPE-Identifier: Automated CPE Identification and CVE Summaries Annotation with Deep Learning and NLP," ChatPaper Journal*, 2024.
[Online]. Available: https://chatpaper.com/chatpaper/paper/22407
[3] Red Hat Research, *"Uncovering CWE–CVE–CPE Relations with Threat Knowledge Graphs," Red Hat Research Publications*, 2023.
[Online]. Available: https://research.redhat.com/blog/publication/uncovering-cwe-cve-cpe-relations-with-threat-knowledge-graphs
[4] MITRE Corporation, *"Common Vulnerabilities and Exposures (CVE)," MITRE CVE Program*, 2024.
[Online]. Available: https://www.cve.org/
[5] FIRST, *"Common Vulnerability Scoring System v3.1: Specification Document," Forum of Incident Response and Security Teams (FIRST)*, 2023.
[Online]. Available: https://www.first.org/cvss/v3.1/specification-document
[6] RiskVision Knowledge Base, *"Vulnerability–CPE Matching," Knowledge Owl Documentation*, 2022.
[Online]. Available: https://riskvision.knowledgeowl.com/help/vulnerability-cpe-matching
[7] NIST, *"National Vulnerability Database (NVD): CPE and CVE Integration Framework," National Institute of Standards and Technology*, 2023.
[Online]. Available: https://nvd.nist.gov/
[8] S. Shin and Z. Xu, *"A Framework for Automated Software Vulnerability Assessment Based on CVE and CPE Correlation," Journal of Information Security and Applications*, vol. 72, p. 103522, 2023.
[Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212623001258