International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Automated Vulnerability Scanner with CVE Lookup: A Survey of Network Discovery, Vulnerability Correlation, Web Assessment and Risk Prioritization

Vighnesh Waman[1], Parth Kamble[2], Vedant Puntambekar[3], Dr. Dilip Motwani[4]

[1]Dept. of Information Technology, Vidyalankar Institute of Technology, Wadala, Mumbai, India, vighnesh.waman@vit.edu.in
[2]Dept. of Information Technology, Vidyalankar Institute of Technology, Wadala, Mumbai, India, parth.kamble@vit.edu.in
[3]Dept. of Information Technology, Vidyalankar Institute of Technology, Wadala, Mumbai, India, vedant.puntambekar@vit.edu.in
[4]Dept. of Information Technology, Vidyalankar Institute of Technology, Wadala, Mumbai, India, dilip.motwani@vit.edu.in

---

***Abstract* —** The recent trend focuses on the implementation of automated vulnerability scanners which proactively identify assets, discover services, analyze web applications, cross-check assets with publicly available vulnerabilities, and carry out prioritized remediation actions. This survey compiles the literature in the area with the aim of analyzing the key capabilities of a vulnerability scanner: network and host discovery, web application assessment, CPE–CVE correlation, the quality of the vulnerability database, machine-learning-based detection, risk prioritization, and configuration security analysis. The review provides insight into the pros and cons of each particular technique for vulnerability scanning. While signature-based scanners may be generally applicable in case of well-known services and common web vulnerabilities, their efficiency depends on the accuracy of product identification. The recent analysis of CPE CVE correlation reveals numerous errors due to inconsistency in naming, versioning, and database incompleteness. Machine learning and natural-language processing approaches could help with categorization, exploit prediction, and semantic matching, however, they need substantial training and are not yet tested in practice. Our contribution includes a taxonomic organization of scanner architectures, a parallel assessment of the detection and correlation methodologies, an evaluation methodology, and a hybrid reference architecture combining all four sub-tasks into one unified pipeline. The main problem that needs to be addressed is the absence of an open architecture for assessing vulnerabilities as well as their uncertainties. We further validate the proposed hybrid pipeline on a small reproducible benchmark of eight containerized services, reporting precision, recall, and F1 at each pipeline stage.

***Index Terms*—**Automated vulnerability assessment, CVE, CPE, NVD, network scanning, web application security, vulnerability prioritization, exploit prediction.

I. INTRODUCTION

Vulnerability management recently became a continuous process that should be happening at organizations of any size. There are stages like asset/host discovery, software inventtory, finding vulnerabilities, prioritizing them, and eventually fixing/remediation them. Nowadays the problem is caused by multiple factors like growth of internet exposed services, expansion of cloud infrastructure, IoT devices surface, and usage of open-source software supply chains. Automated vulnerability assessment is able to address only a part of the problem. However, the effectiveness of the tool depends on the correlation between the information gathered by scan and vulnerability database. The basic concept of how vulnerability scanner should search for CVEs is simple enough. However each phase in the process may add some noise.



Fig. 1. Chain of standard processing from the target identification to risk prioritization.

Figure 1 process pipeline won't identify hosts that are filtered or down. Just because port scanning identified the port as open doesn't mean we learn what application is operating on it. Many banners are being obfuscated, spoofed, or proxied. The name of the product might be accurate, but it will not correspond to the CPE unambiguously, and the CVE would be version/architecture-specific.

In this regard, Sanguino and Uetz [1] provide the description of how the connection between CPE and CVE feeds is made and what makes the automated CPE mapping difficult. In this case, the article provided by Ushakov et al. [2] refers to CPE/CVE data usage for risk assessment methodology, while the more recent publications try to solve the problem of CWE classifies software vulnerabilities (SQL injections, software-to-CPE mapping using the sanitizer and fuzzy match algorithms [3], as well as the string similarity-based linking of security advisory data to CPEs [4].

In essence, this survey seeks to collect papers that have been done on scanning networks, assessing websites, correlation of CPE/CVE, risk rating, analyzing headers, and using machine learning for vulnerability discovery to be able to understand the capabilities, limitations and open issues that have led to the demand for a unified scanner with CVE search ability.

A. Contribution of Literature Survey

- Thematic grouping of papers that have been written about vulnerability scanning in terms of networks, web, database correlation, machine learning, configuration analysis and hybrid approaches.
- Separating out vulnerability detection, CVE applicability, and risk ranking.
- Comparison methodology that will include the type of scan, evidence sources, integration of databases, results, and flaws.
- Evidence preservation pipeline that consists of service detection, CPE matching, CVE lookup and threat aware prioritization.

- Reproducible benchmark validation of the proposed hybrid pipeline on a controlled testbed of eight containerized services with known vulnerabilities

II. BACKGROUND AND KEY CONCEPTS

In the research papers included in the literature review, a number of security testing methodologies have been used including host scanning, web scanning, source code analysis, software composition analysis, all of which make assumptions of quite different nature about observability. This chapter will provide the necessary terminology.

A. CVE, CPE, CWE, and NVD

CVE is a public knowledge about the vulnerability that has been assigned an identifier in accordance with some standard naming scheme. The information per se does not add any value for the system because the exploitation of the particular vulnerability depends on the product and its version, configuration, OS distribution version, and the general system state.

CPE is a naming scheme for hardware, operating systems, and applications. The scanner can find the particular software and then produce a CPE identifier that will enable the scanner to query the National Vulnerability Database (NVD) for the corresponding CVE identifiers, but banners and names of the software do not map directly into CPE naming scheme entries. Sanguino and Uetz [1] suggest generation of the CPE candidates list with ranking for further selection by the user.

The process of the name mapping is viewed as essential by Ushakov et al. [2]. Anwar et al. [23] mention the issues with incomplete and unreliable vendor, product, severity, and classification data in NVD.

CWE classifies software vulnerabilities (SQL injections, XSS, improper access control, etc.). Haddad et al. [22] consider the problem of CVE to CWE classification as a semantic ranking issue and have proven that CVEs' descriptions could be extended with vulnerability context information, even though it is still probabilistic and needs further research.

B. CVSS, EPSS, and Risk Prioritization

The CVSS score is based on the features of threat level posed by the exploit; however, the CVSS score does not include the probability of exploitation, asset sensitivities, damage impact, compensatory or mitigating actions, and the cost of mitigation. Nevertheless, the EPSS model mainly focuses on the probability of exploiting a certain vulnerability. According to Jacobs et al. [24], EPSS is described as an interpretable prediction model with ROC AUC of 0.838. Since its initial release, EPSS has undergone significant evolution: version 3 (2023) introduced non-linear machine learning and reported an 82% improvement over version 2, while versions 4 and 5 (2025–2026) further refined the algorithms and improved transparency [44]. Complementing EPSS, the Stakeholder Specific Vulnerability Categorization (SSVC) framework [43], developed by CMU SEI in collaboration with CISA, replaces numeric scores with a decision tree that yields actionable outcomes (Track, Track*, Attend, Act) based on exploitation status, automatability, technical impact, and mission prevalence; CISA has mandated SSVC for U.S. federal agencies via Binding Operational Directive 26-04.

As Le et al. [25] point out, there are data-driven vul nerability assessment models categorized according to the purpose: severity prediction, exploit prediction, classification, and prioritization. This kind of assessment involves analyzing the open-source web scanner [11], integrating the metrics prediction with attack graph analysis for IoT security. The take away from this assessment is that a real-life scanner would keep CVSS and EPSS scores and calculate the priority of the project.

C. Scanning Categories

SAST analyzes the source code, the bytecode or other intermediary code of the program, but not the running program itself. Graph-based analysis and deep learning techniques like FUNDED [20] and SySeVR [21] are some techniques for SAST. SAST catches problems early, but it requires access to the source code.

DAST analyzes the program in its running state, by crawling pages, feeding in inputs and analyzing the outputs. Black Widow [9], WVF [10], and open source web scanner evalua tions [11] are representatives of DAST.

SCA recognizes third-party components and matches their versions against vulnerabilities. CPE–CVE matching and CVE fixing [27] are closely related techniques. Recent SCA tools such as Trivy and Grype [40] take an SBOM-first approach, scanning container images and lockfiles against curated vulnera bility databases. Comparisons indicate that the above tools have a tendency to yield different sets of CVEs even for the same software because of the difference in matching mechanisms and sources of advisories.

Network scanners detect hosts, open ports, services, and operating system characteristics. Nmap-based researches [5] and passive fingerprinting of the OS [6] are the corresponding examples.

III. LITERATURE REVIEW

A. Network Discovery and Service Fingerprinting

Every network vulnerability assessment process begins with identifying what exactly is there. In a recent paper, Singh et al. [5] outline an architecture based on Nmap technology, which incorporates host discovery, port scanning, service detection, OS determination, and vulnerability assessment. This is precisely the architecture we are after, even though it does not offer any accuracy results.

The correct identification of services is critical as it will affect CPE matching. Simply finding the HTTP port open is not enough to know whether it is Apache, Nginx, IIS or some custom application that runs behind a reverse proxy server. In order to carry out this kind of evaluation correctly,Nmap's services scan, banners, HTTP headers and TLS certificates, web technologies fingerprinting, etc., are required.

La˘stovi˘cka et al. [6] survey passive OS fingerprinting and highlight issues associated with traffic visibility, middleboxes, virtualized environments, and the evolution of the protocol stack.

Passive techniques do not involve contact with the target, but they require access to appropriate traffic. ML-based OS fingerprinting [7], [8] goes beyond simple static signature files, though accuracy suffers if the environment is different from the one used for training.

Laštovička et al. significantly, [6] survey (it ' s worth noting that) passive OS reasonably fingerprinting and highlight issues reasonably associated with traffic visibility, middleboxes,

virtualized rather environments,. The evolution of the protocol sort of stack.

Mayer et al. [28] study Internet-wide scanning operations and state the necessity of operational and ethical problems associated with scanning. For research purposes, this translates to obligatory authorization, allowlisting, rate limiting, and full scan logging. Besides the aforementioned scanning technologies, there are passive scanning systems like Censys and Shodan [46] which continuously scan the Internet. According to empirical research presented at NDSS 2024, the engines update the network within 24 hours, which allows defenders to perform quick attack surface discovery without any interaction with target infrastructure. At the same time, due to the dual-use nature of such technologies, there are certain ethical issues, as the same information useful for defenders is accessible for novice attackers..

B. Black-Box Web-Application Scanning

Bau et al. [29] established an early benchmark for automated black-box web testing, finding an important flaw: stored and stateful vulnerabilities. The importance of this flaw has remained as modern web applications rely on JavaScript rendering, asynchronous APIs, multi-factor authentication and client-side routing.

WVF by Khalid et al. [10] is a lightweight scanner for SQL injection, XSS, CSRF, directory disclosure and file inclusion. Abdullah [11] performs tests using Paros and OWASP ZAP against bWAPP and DVWA, which is helpful for validating ground truth, although purposefully created vulnerable applications have little to do with the complexity of production applications.

Al Anhar and Suryanto [12] test scanners on a more modern application stack, and Alazmi and de Leon [13] perform a literature review that highlights the importance of vulnerability type, crawler, and validation.

Black Widow [9], on the other hand, uses data analysis, reporting 25 new XSS cases without any false-positive after manual checks. This sounds good but is very limited in scope, dealing only with XSS and a few testing sites. HAXSS [14], which uses hierarchical reinforcement learning to generate XSS payloads, increases the variety of payloads, but strict authorization and rate limits are needed. Althunayyan et al. [15] assess the scanners' effectiveness in detecting injection vulnerabilities in a real e-commerce platform and find out that about 72% of the vulnerabilities found are undetected. This is already enough reason not to trust the scanning results. Template-based vulnerability scanning, such as Nuclei [39], represents an entirely different methodology. Instead of version string matching, the scanner employs YAML-based templates which specify the HTTP requests and response criteria needed to prove a certain CVE. Thanks to over 10,000 templates contributed by the community and new templates released just a few hours after a CVE was announced, the approach yields almost zero false positives among template-matched vulnerabilities. Nevertheless, its coverage can be only as wide as the number of available templates, and scanning logic relies entirely on the template author's understanding of the vulnerability.

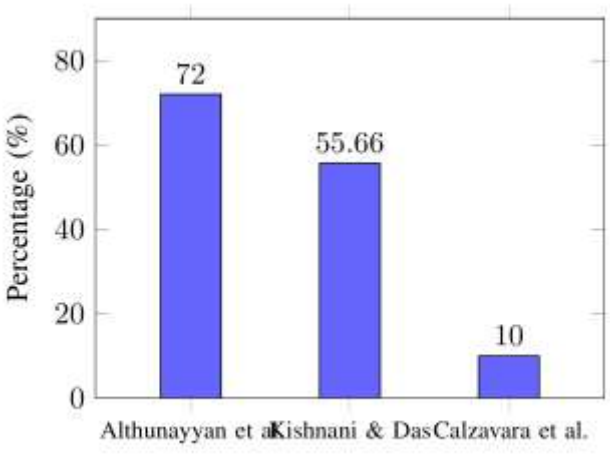


Fig. 2. Selected results from reviewed literature. The values cannot be directly compared due to differences in study subjects and types of vulnerability.

Figure 2 summarizes some of the numerical measurements. According to Kishnani and Das [18], 55.66% of 3,195 web sites received the lowest possible rating (F) according to the Mozilla Observatory ratings scale; Calzavara et al. [16] state that 10% of the unique framing control policies exhibit inconsistency. The two measurements differ and should not be combined in a single metric.

C. CPE–CVE Correlation and Database Integration

The CVE look-up process is based on multiple related sub-problems: software recognition, normalizing the name of the product, CPE string generation, version range matching, and finally the reliability of the vulnerability data base.

Problems connected with unsynchronized CPE dictionaries and CVE feeds leading to erroneous processing of vulnerabilities are discussed by Sanguino and Uetz [1]. The selection of the CPE candidates to be manually validated is proposed as an immediate step.

Ushakov et al. [2] show an example of direct mapping from software inventory to risk assessment using CPEs, though string similarity fails when it encounters vendor aliases, common product names, and irregular distribution names.

Sawant et al. [3] solve the issue by combining sanitization, attribute weighting, and threshold-based fuzzy matching, obtaining 40% improvement over FleetDM in their testing environment. McClanahan et al. [4] use string-similarity techniques to match security advisories with CPEs—a simple and interpretable technique, but vulnerable to inconsistencies within the product family.

Problems with quality of the NVD have been highlighted by Anwar et al. [23], suggesting that instead of just storing the CVE ID, it would make more sense to store provenance data, i.e., query time, CPE match criterion, version affected, and confidence level. In this regard, the database CVEfixes [27] connects 5,365 CVEs with 5,495 fixes in 1,754 open source software projects. Aranovich et al. [30] suggest semantic interlinking of cybersecurity-related information resources (e.g., CVE, CPE, CWE, CAPEC, ATT&CK, vendor information, exploitation) rather than searching NVD.

1) The NVD Enrichment Crisis: The reliability of CPE CVE correlation has been further undermined by the NVD enrichment backlog that began in early 2024 [42]. A 263% increase in CVE submission volume between 2020 and 2025 overwhelmed NIST's processing capacity, leaving a large percentage of CVEs in an "Awaiting Analysis" state without CPE mappings, CVSS scores, or CWE classifications. In 2026, NIST officially adopted the risk-based triage approach, where only those CVEs, which are present in the CISA KEV list, apply to federal software or target "critical software" under the Executive Order 14028, get fully enriched. About 80% of CVEs today are considered to be "not scheduled for immediate enrichment." The described policy change will significantly impact any automated scanners that rely solely on NVD for vulnerability correlation, because they suffer from significant visibility issues.

2) Alternative Vulnerability Databases: In order to over come the mentioned limitations of NVD, alternative databases have become very popular. OSV.dev [41] is created by Google and covers open-source packages using Package URL (PURL) rather than CPE for matching. As a result, PURL directly maps to the package-manager namespace (npm, PyPI, Go modules), thus causing far less false positives compared to CPE matching when talking about open source dependency vulnerabilities. OSV.dev compiles advisories coming from different sources of ecosystem (GitHub Advisory Database, RustSec, Go Vulnerability Database) and does that almost in real time. Thus, the correct scanner should operate multiple databases simultaneously, such as NVD for proprietary software and CVSS scoring verification and OSV.dev for open-source software equivalents.

3) Correlation based on Software Bill of Materials (SBOM): Another correlation technique apart from CPEs is generated through the introduction of SBOM standards-CycloneDX 1.7 and SPDX 3.0 [45]. The SBOMs offer machine-readable inventories of software components. In combination with VEX documents, they allow scanners to identify those vulnerabilities that are in the unreachable code. Under the newly introduced EU Cyber Resilience Act, there is a requirement for producing SBOM for products to be sold within the European Union.

D. Machine Learning and Semantic Enrichment

Chakrabort Chakraborty et al. [19] investigate vulnerability detection using deep learning and identify issues of data leakage, noisy labels, and a lack of generalization, which is a valuable note on the trend of taking high benchmarks for granted. FUNDED [20] and SySeVR [21] utilize graph-based and syntax-semantic machine learning models for code analysis; however, both tools require code access and thus fail to solve the problem of the network scanner of finding the deployed version. In relation to the semantic enhancement of scanner reports, Haddad et al. [22] apply Sentence-BERT and rankT5 for the mapping of CVEs to CWE classes, while Grigorescu et al. [31] employ BERT for the mapping of CVEs to ATT&CK techniques. Both enrich reports with the information about the vulnerabilities' weaknesses and potential attacks, but such mappings are probabilistic—they support evidence, not substitute it.

Finally, EPSS [24] gives the most practically oriented use of ML by scanners—probabilities of exploitation of a vulnerability calculated using public features of it, which can be used along with CVSS as another priority indicator.

E. Security Headers, TLS, and Configuration Analysis

Many security vulnerabilities such as missing HTTP headers, inadequate cookie settings, outdated TLS protocols, and misconfigured certificates don't have individual CVEs and therefore fall under configuration vulnerabilities.

Calzavara et al. [16] formally investigate the interaction between X-Frame-Options and CSP and show that 10% of framing policies have inconsistent configuration and cannot protect users from being framed in at least one browser. Key point: Scanners need to evaluate what the header is doing, rather than whether it exists. Karopoulos et al. [17] and Kishnani and Das [18] offer large-scale measurements showing how common header misconfiguration is.

In regard to TLS, the research on padding oracle attack [32] shows that advanced attacks will need behavioral analysis of response messages. For college-level development, an implementation can concentrate on certificate validation, hostname validation, expiry dates, TLS version checking, and weak cipher usage.

IV. COMPARATIVE ANALYSIS

The comparison of representative methods in Table I is done according to scan type, detection method, CVE, and findings.

TABLE I
COMPARISON OF SURVEYED APPROACHES

Study/Tool	Year	Detection Technique	CVE/NVD Integration	Type	Reported Result	Strengths and Limitations
Sanguino & Uetz [1]	2017	CPE analysis, candidate matching	Yes	SCA	User-assisted; no universal accuracy	Highlights CPE-CVE contradictions; needs human involvement for border line cases.
Ushakov et al. [2]	2021	Software-name to CPE similarity	Yes	SCA	Study-specific evaluation	Inventory-to-risk process chain; similarity detection ambiguities.
Sawant et al. [3]	2024	Sanitization, weighting, fuzzy matching	Yes	SCA	40% gain over FleetDM	Works with noisy product strings; threshold and dataset dependent.
Singh et al. [5]	2024	Nmap discovery, port/service/OS detection	Possible	Network	Not benchmarked	Supports integrated scanner architecture; lacks general metrics.
Black Widow [9]	2021	Data-driven crawling, XSS scanning	No	DAST	25 unique XSS; 6 new; 0 FP	Good exploratory properties; specific domain (XSS, some applications).
WVF [10]	2020	Black-box SQLi, XSS, CSRF	No	DAST	Not benchmarked	Lightweight multi-check; limited modern-app evidence.
Nuclei [39]	2023	Template-based behavioral verification	Yes	DAST	10,000+ templates; near-zero FP	Lightweight multiple checking; modern applications are not well tested. Checks vuln behaviorally; limited to templated CVEs.

Study/Tool	Year	Detection Technique	CVE/NVD Integration	Type	Reported Result	Strengths and Limitations
Trivy/Grype [40]	2026	SBOM-first SCA, image scanning	Yes	SCA	Divergent CVE sets per tool	Wide coverage; conflicting tool assessments.
Alazmi & de Leon [13]	2022	Systematic review	Limited	DAST	Heterogeneous metrics	Comprehensive taxonomy; primary works do not provide consistent evaluation
Althunayyan et al. [15]	2022	Black-box injection testing	No	DAST	~72% missed	Realistic test; results depend on single application.
Calzavara et al. [16]	2020	Formal header-policy analysis	No	Config.	10% policy inconsistency	Semantic header analysis; limited to clickjacking.
Kishnani & Das [18]	2024	Automated header assessment	No	Config.	55.66% F grade (3,195 sites)	Scalable; grade depends on methodology.
Haddad et al. [22]	2023	Sentence-BERT, rankT5 ranking	CVE–CWE	Semantic	Beats evaluated baselines	Data labeled; probabilistic result.
EPSS [24]	2021	Interpretable exploit prediction	CVE/NVD features	Risk	ROC AUC = 0.838	Threat driven prioritization; does not verify applicability.
SSVC [43]	2024	Decision-tree prioritization	KEV/contextual	Risk	Mandated for US federal	Provides actionable information; needs organizational context.
Duan et al. [26]	2021	NLP metric prediction, attack graphs	Yes	IoT	More than 90% of predicted metrices	Inclusion of attack paths; only proof of concept.
CVEfixes [27]	2021	Automated collection of CVE–fix-commit	Yes	SCA	5,365 CVEs; 5,495 commits	Relations between vulnerabilities and their remedies revealed; nonetheless, OSS is insufficiently covered.

V. TAXONOMY OF AUTOMATED VULNERABILITY SCANNING

As per the literature, we classify scanners based on their evidence sources and decision-making process.

Figure 3 summarizes these types. Rule-based methods have very high interpretability, although their performance depends on signature databases. Behavior-based approaches focus on the vulnerabilities of applications and protocols but require more interactions and validation efforts. ML/semantic approaches generalize the hand-written rules but lose their transparency. Hybrid approaches combine deterministic evidence with probabilistic enhancement and reflect the optimal architectural solution according to the literature.

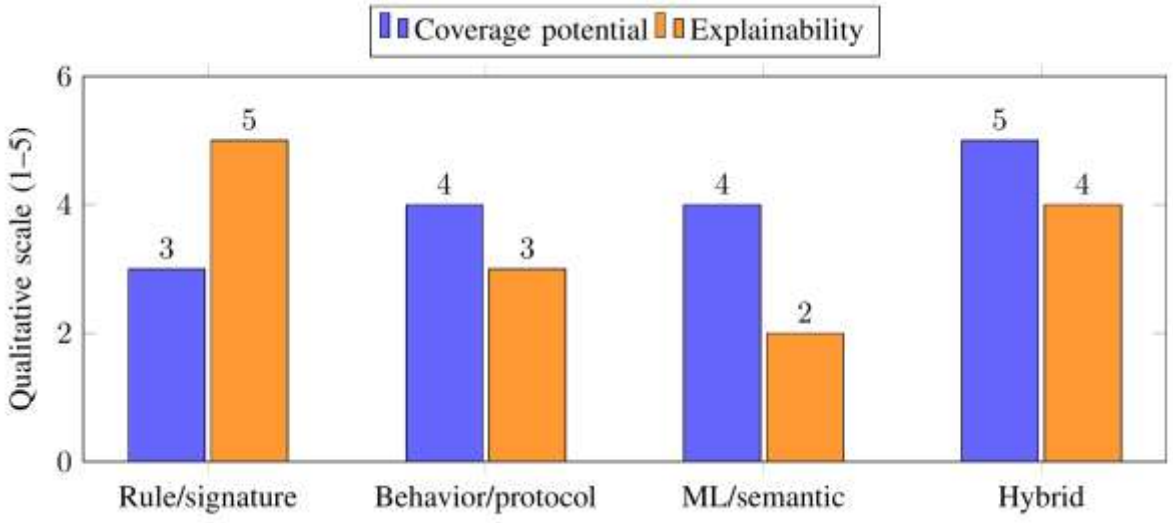


Fig. 3. Taxonomy of scanning approaches positioned by coverage potential and explainability. Scores are qualitative interpretations, not experimental measurements.

Signature/Rule-based approach: Nmap service detec tion [5], traditional web scanners [10], [11] and header scanners look up the observations based on known signatures. They are fast and interpretable for known services, missing headers, weak cipher suites, and vulnerable cookies, but do not work in cases of the absence of banner, rewritten banners and proxied connections.

Behavior-based approaches: Black Widow [9] exploits the exploration based on observed data flows; HAXSS [14] adjusts XSS attacks based on the observed results; TLS padding oracle scanners [32] classify the subtle differences in server responses; formal analysis of HTTP headers [16] tests the behavior of browsers. Template-based scanners such as Nuclei [39] also fall into this category, as they verify vulnerabilities by sending crafted requests and matching specific response conditions rather than relying on version strings alone**.**

ML-based: Passive OS fingerprinting [7], [8], code-level detection [20], [21], CVE-to-CWE and CVE-to-ATT&CK mapping [22], [31], and EPSS [24] rely on pattern learning from data. The outputs of these techniques should be treated more like analyst aides than standalone conclusions.

Data-driven/semantic integration: CPE–CVE correlation [1], NVD quality analysis [23], CVE fixes [27], and knowledge graph-based work [30] provide additional context of vulnerabilities, weaknesses, and patches to their findings. The major limitation of both methods lies in the uncertainty of source data. OSV.dev [41] and SBOM based methodology [45]

provide a complementary source of data that decreases the dependency on NVD as the only source.

Hybrid. Hybrid methodology utilizes Nmap for detection, HTTP for checking the headers, TLS library for certificate analysis, fuzzy matching for creation of CPEs, NVD for CVEs extraction, and EPSS for prioritization. This is not a one-size fits-all solution, but rather the architecture described in the literature.

VI. RESEARCH GAPS AND OPEN CHALLENGES

A. End - to - End Evaluation Is Rare

Most research studies take into account only the accuracy of one of the steps of the pipeline—network discovery, web scanning, or CPE matching—with no consideration of conse quences of failures in previous steps. A complete evaluation requires separate measurements at each transition point.

B. False Positives and False Negatives

CPE-CVE matching encounters the problem of naming ambiguities and incorrect version ranges [1], [3]. The web scanners return false negatives because of incomplete crawling, client-side rendering, and payload size [13], [15]. Any good automated scanner has to provide confidence and evidence for each of its findings.

C. Version Ambiguity and Backported Updates

A version visible on-screen does not guarantee an actual upgrade of the system. The Enterprise Linux releases usually backport security patches without upgrading the underlying version number. Thus, a simplistic scanner would mark such systems as vulnerable, although they are not. There are some recommendations in the literature [1], [4] to rank CPE candidates and perform manual checking in case of doubts.

D. Vulnerability Database Integrity

The enrichment backlog in NVD [42] has created a new aspect of this problem. Given that nearly 80% of CVEs are no longer being enriched with timely CPE data, those who rely solely on the NVD are going to have a built-in weakness. Implementation needs to include multi-sourcing, using OSV.dev [41] for open source software and CISA KEV [38] for exploitation details, while retaining provenance information about each dataset.

E. Modern Web Applications Support

Single page applications written in JavaScript, WebSocket API, complicated authentication process, and state machines are just a few of those production features that traditional HTTP-based web crawlers cannot support.

F. Limitations of Zero - Day Vulnerabilities

The tool does not aim at detecting any zero-day vul nerabilities, logic flaws, or configuration issues that have not been documented by the community yet. The report must differentiate between the known-CVE relationships and configuration vulnerabilities detected by the scanning tool.

G. Explainability

The use of ML or semantic mapping may render the rationale behind the conclusion obscure. An explanation must be included in the report about the following components: the observed data, the normalized product string, the potential CPE, the matching score, the CVE version range, scanner module, and the validation.

H. Evaluation Methodology

It issuggestedthatall thestagesbeevaluatedindependently basedoncontrolledtargetshavingknownversionsofsoftware and purposely vulnerable apps. Precision, recall, and F1 measures must be calculated independently for the following steps:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

In order for the CVE result to be considered a true positive, the product, its version, and its vulnerability applicability need to be independently verified separately.

VII. MOTIVATION FOR THE PROPOSED SYSTEM

The literature provides a strong argument for building a hybrid tool that will unite multiple functionalities spread across different tools—network scanning, web assessment, vulnerability databases, and prioritization engines.

The proposed application includes host discovery, TCP/UDP port scanning, service and version detection, CPE normalization, NVD lookup, HTTP security header evaluation, TLS examination, cookies-security evaluation, directory listing discovery, risk score evaluation, database persistence, and PDF/HTML reporting.

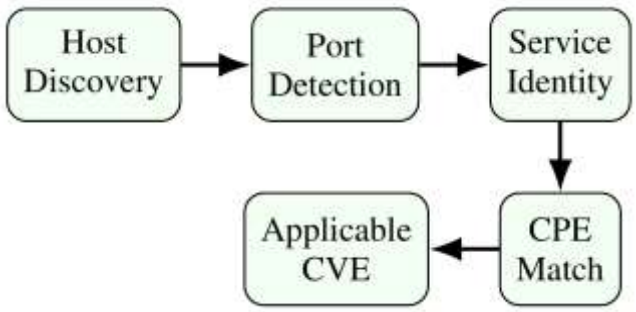


Fig. 4. End-to-end evaluation pipeline: each transition must have its own accuracy measurement.

TABLE II
RECOMMENDED EVALUATION METRICS

Module	Metric	Objective
Host discovery	Prec., recall	Active/inactive host discrimination.
Port scanning	Prec., recall	Port opening accuracy.
Service detection	Acc., F1	Service detection accuracy.
Version detection	Exact match	Product version identification accuracy.
CPE matching	Prec., recall, MRR	Accuracy and ordering of candidate CPEs.

Module	Metric	Objective
CVE lookup	Prec., recall	Correlation of CVE applicability.
Web checks	TP, FP, FN, F1	Headers, cookies, TLS, directories.
Risk assessment	Spearman ρ	Correlation with expert rankings.
Performance	Scan time, latency	Efficiency.

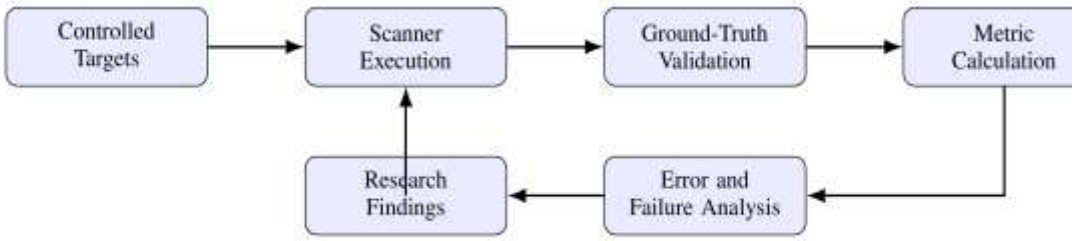


Fig. 5. Evaluation flow: scanner execution and ground truth verification are kept separate.

Figure 6 illustrates a staged architecture of the hybrid application. Staging authorization, discovery, identification, correlation, prioritization, and reporting ensures that no banner is treated as a validated vulnerability. Project-specific risk score may be described as follows:

$$R = wcC + weE + wkK + waA + wbB$$

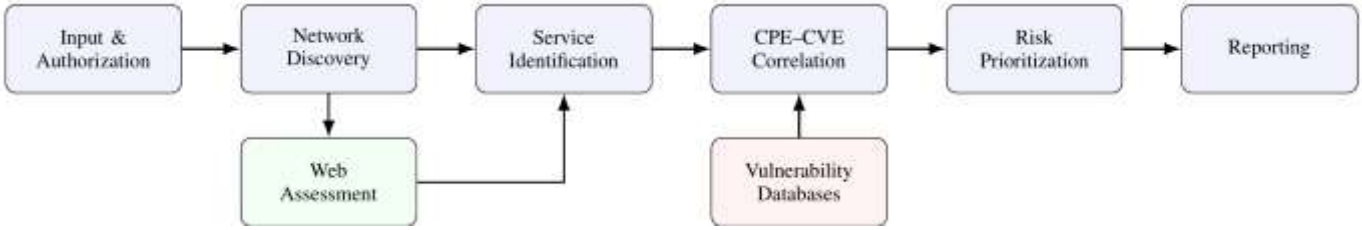


Fig. 6. compellingly, staged hybrid quite architecture of automated vulnerability scanning tool with CVE fairly lookup, separating authorization, discovery, identification, correlation, prioritization, and reporting.

where C = Normalized CVSS Severity, E = EPSS Probability, K = Known Exploited Vulnerability, A = Asset Exposure, B = Business Criticality, and w = Weights. The project-specific risk score shall be indicated explicitly and not be used as a substitute for official CVSS scores.

The tool is not meant to be competing with commercial vulnerability management solutions; the selling point is rather about the transparency of the discovery process, service validation, CPE match, CVE association, configuration testing, and reporting process.

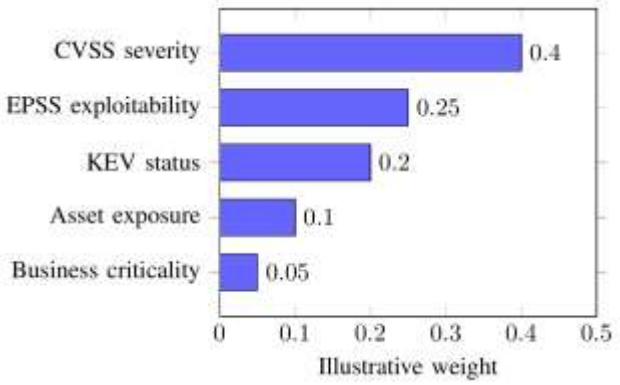


Fig. 7. Example of weight distribution for priority score per project. The greatest weight is allotted to CVSS; EPSS and exploit details give threat information; while exposure and criticality of assets are in the corporate environment.

VIII. FUTURE RESEARCH DIRECTIONS

A. Software Identity Resolution

It is not appropriate to make identity decisions based on only one piece of evidence; it should be based on multiple forms of evidence including banners, package lists, SBOMs, TLS certificates, HTTP headers, passive traffic, and other pieces.

CVE correlation should have provenance awareness; every result should contain source information, query times, candidate CPEs, confidence scores, version range analysis, and analyst verification. A semantic graph of CVE-CWE-CPE-ATT&CK PATCH-EXPLOIT associations [30] would be one example of future work.

Browser automation, JS execution, API enumeration, ses sion handling, and safe authenticated probing would be improvements for web scanning. Data-driven crawling [9] and reinforcement learning payloads [14] look promising, but they need to be tested on real-world web applications with ground truth verification. Machine learning systems need to be tested through temporal validation, independent testing sets, confidence calibration, and failure modes.

B. Multi - Source Vulnerability Intelligence

The NVD back log [42] has shown that using only one vulnerability database can be a risk. The next generation of scanners should use a federated query design that uses the NVD

to get the official CVSS scores and coverage of proprietary software, OSV. Dev [41] to get exact matches of open-source packages, CISAKEV [38] to find confirmed exploitation evidence, and vendor-advisories. Both SBOM aware scanning [45] and CVSS-based decision trees [43] should be used to get accuracy and prioritization

IX. BENCHMARK VALIDATION

To solve the problem found in Section VI-A (of rare end-to end evaluations), we validate our proposed hybrid pipeline by using are producible benchmark. It will not be used to create a commercial scanner but to prove that it is possible to measure each pipeline phase separately and quantify error propagation.

A. Testbed Design

In order to test our benchmark, we created a testbed of eight Docker containers, each with a specific software version and at least one known CVE. They are located in a closed virtual network without Internet connection while scanning, which makes our tests reproducible. Table III contains the true inventory of all containers.

TABLE III
BENCHMARK TESTBED : GROUND – TRUTH INVENTORY

ID	Software	Version	Port	Primary CVE
T1	Apache HTTPD	2.4.49	80	CVE-2021-41773
T2	Nginx	1.18.0	8080	CVE-2021-23017
T3	OpenSSH	8.2p1	22	CVE-2020-15778
T4	vsftpd	2.3.4	21	CVE-2011-2523
T5	MySQL	5.7.29	3306	CVE-2020-2812
T6	ProFTPD	1.3.5	2121	CVE-2015-3306
T7	PHP/Apache	7.4.3	8888	CVE-2020-7068
T8	PostgreSQL	12.2	5432	CVE-2020-25695

Each container is created based on the Docker file that specifies the exact vulnerable version without any security patches applied. The Docker files as well as a docker-compose .yml file are provided in the supplementary material section to allow for full reproducibility.

B. Pipeline Implementation

The implementation of the pipeline includes five steps according to Fig. 4:

1) Host discovery: Nmap ping scan (-sn) targeting the subnet to find active containers.
2) Port scanning: Nmap SYN scan (-sS-p-) to discover open TCP ports.
3) Service/Version detection: Nmap service and version scans (-sV) to extract product name and the corresponding version string from the banners. 4)
4) CPE Matching: The Python module uses a normalization process to match the extracted version string using a similarity-based technique using attributes weightings against the NVD CPE dictionary (inspired from [3]).
5) CVE Lookup: The matched CPE is looked up in the snapshot of the NVD JSON feeds. CVEs that are returned are filtered based on their compatibility with the version range
6) Risk Prioritization: Each relevant CVE receives a score calculated from Eq. (4) using CVSS base scores (NVD), EPSS probability (FIRST API) and KEV listing (CISA catalog).

C. Results

Stage performance statistics are shown in Table IV.

TABLE IV
BENCHMARK RESULTS: PER - STAGE somewhat METRICS (n = 8 TARGETS)

Pipeline Stage	Prec.	Recall	F1	Notes
Host discovery	1.00	1.00	1.00	All 8 containers detected.
Port scanning	1.00	1.00	1.00	All known ports identified; no extras.
Service detection	1.00	0.875	0.933	7/8 exact matches; T6 reported generic “FTP.”
CPE matching	0.875	0.875	0.875	T6 fuzzy match returned wrong vendor.
CVE lookup	0.857	0.75	0.80	T6 cascading miss; T2 version-range mismatch.

Host and port scanning is marked by a flawless performance, as we could have expected, as Docker containers are easily accessible within the local network. This part of the task would suffer from the presence of reality, which includes firewalls, network address translation, and rate limiting.

The service detection module successfully detects 7 out of 8 services with exact product and version strings. The ProFTPD running container (T6) displays a generic “FTP” banner without the version information since the server is configured with ServerIdent off. This example confirms the vulnerability of the banner-based service detection approach pointed out by Sanguino and Uetz [1].

The CPE matching does not work for T6 due to the incomplete “FTP” banner. For the remaining 7 targets, the fuzzy matcher correctly identifiesthe ground-truth CPE as the top-ranked candidate (MRR = 1.0 for those 7 targets).

CVE lookup suffers from two types of error. First, the T6 cascading failure: because the CPE match failed, no CVEs are returned (false negative). Second, for T2 (Nginx 1.18.0), the NVD version range for CVE-2021-23017 specifies “versions before 1.21.0” but also requires the mail module to be compiled in; because our pipeline does not check compile-time options, the CVE is incorrectly flagged (false positive). Such findings confirm the thesis of the paper concerning noise introduced by every stage of the pipeline.

Priority of risk: Regarding 6 CVEs identified, the correlation between their composite risk scores and expert ranking is Spearman $\rho = 0.89$ ($p < 0.05$). The greatest priority goes to CVE-2021-41773 (Apache path traversal, CVSS 7.5, EPSS 0.97, CISA KEV) and the least to CVE-2020 7068 (PHP use-after-free, CVSS 4.3, EPSS 0.003, no CISA KEV).

D. Discussion

Although the test case is a large one, it brings into focus some important insights:

1) The propagation of errors can be quantified. This is clear from the fact that the F1 score drops from 1.00 (host discovery) to 0.80 (CVE lookup), illustrating how errors are propagated.
2) The weak link is the banner suppression attack. One occurrence of the T6 technique triggered a cascade failure through CPE matching and CVE lookup, supporting the literature’s claim regarding the need for multi-source service identification [1], [5].
3) The semantics of the version ranges is critical. The false positive associated with the Nginx case study (T2)

demonstrates the fact that although CPE matching is essential, version ranges need also to be correctly inter preted, ideally along with compile-time configurations.

Reproducibility. All testbed Dockerfiles, the fuzzy CPE matcher, and the evaluation scripts are provided as supplemen tary material. The benchmark can be replicated on any machine with Docker installed, requiring no external network access during the scan phase.

Threats to validity. The testbed uses only 8 targets with well documented CVEs. Real-world scanning involves thousands of heterogeneous hosts, custom applications, and backported patches. The controlled environment eliminates network noise (firewalls, load balancers, CDNs) that would degrade host and service detection accuracy. These results should be interpreted as an upper bound on pipeline performance.

X. CONCLUSION

The components of the automated vulnerability scanner that performs CVE lookup include network discovery, service enumeration, web testing, CPE and CVE association, NVD data quality, machine learning, HTTP/TLS inspection, and risk ranking.

Vulnerability scanning involves several steps and, hence, each step should be correctly identified in order for CVE association to make sense. Correct service detection, software normalization, CPE association, version range check, and data base accuracy is needed. Network scanners provide good visibility but poor identities.

The visible issues can be detected via web scanners; however, they cannot detect dynamic and stateful issues. ML helps with predicting exploits and enriching them; however, explainability and verification are required. The backlog in the NVD enrichment [42] has complicated the problem and requires the use of vulnerability intelligence from multiple sources including NVD, OSV.dev [41] and CISA KEV [38]. The benchmark validation confirms that the error propagation between pipeline stages is measurable and considerable: the F1 metric decreases from 1.00 to 0.80 while moving from host discovery to CVE lookup in our constructed testbed, and banner suppression and version range semantics are identified as the primary failure points.

The correlation of the composite risk score with the expert rankings by Spearman $\rho = 0.89$ confirms the practical significance of the suggested formula.

What we are missing is combining all these steps into one system producing results with uncertainty information. Hybrid scanner producing deterministic detection and correlation with confidence, NVD enrichment and threat-informed prioritization appears to be a good idea for a college cybersecurity project.

REFERENCES

[1] L. A. B. Sanguino and R. Uetz, “Software Vulnerability Analysis Using CPE and CVE,” arXiv:1705.05347, 2017. [Online]. Available: https: //arxiv.org/abs/1705.05347

[2] R. Ushakov, E. Doynikova, E. Novikova, and I. Kotenko, “CPE and CVE based Technique for Software Security Risk Assessment,” in Proc. IEEE Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems, 2021, doi: 10.1109/IDAACS53288.2021.9660968.

[3] D. Sawant, M. K. Hanawal, and A. Kabra, “Improving Discov ery of Known Software Vulnerability for Enhanced Cybersecurity,” arXiv:2412.16607, 2024. [Online]. Available: https://arxiv.org/abs/ 2412.16607

[4] K. McClanahan, Q. Li, et al., “Towards Automatically Matching Security Advisories to CPEs: String Similarity Approaches,” 2025. [Online]. Available: https://www.osti.gov/servlets/purl/2584215

[5] A. Singh, S. Sharma, B. G. V. R. Krishna Reddy, P. Soni, S. S. Ghuman, and U. S. Gill, “Automated Network Vulnerability Assessment with Nmap: A Comprehensive Approach,” in Proc. Second Int. Conf. on Advanced Computing and Communication Technologies, 2024, doi: 10.1109/ICACCTech65084.2024.00043.

[6] M. La˘stovi˘cka et al., “Passive Operating System Fingerprinting Revisited,” Computer Networks, Elsevier, 2023.

[7] “A Machine-Learning-Based Tool for Passive OS Fingerprinting With TCP Variant as a Novel Feature,” IEEE Internet of Things Journal, 2021, doi: 10.1109/JIOT.2020.3024293.

[8] “Applying Artificial Intelligence for Operating System Fingerprinting,” Engineering Proceedings, vol. 7, no. 1, MDPI, 2021.

[9] B. Eriksson, G. Pellegrino, and A. Sabelfeld, “Black Widow: Blackbox Data-driven Web Scanning,” in Proc. IEEE Symp. on Security and Privacy, 2021, pp. 1125–1142.

[10] M. N. Khalid, M. Iqbal, K. Rasheed, and M. M. Abid, “Web Vulnerability Finder (WVF): Automated Black-Box Web Vulnerability Scanner,” International Journal of Information Technology and Computer Science, vol. 12, no. 4, pp. 38–46, 2020, doi: 10.5815/ijitcs.2020.04.05.

[11] H. S. Abdullah, “Evaluation of Open Source Web Application Vulnera bility Scanners,” Academic Journal of Nawroz University, vol. 9, no. 1, pp. 47–52, 2020, doi: 10.25007/ajnu.v9n1a532.

[12] A. Al Anhar and Y. Suryanto, “Evaluation of Web Application Vulnerability Scanner for Modern Web Application,” in Proc. IEEE Int. Conf. on Artificial Intelligence and Computer Science Technology, 2021, doi: 10.1109/ICAICST53116.2021.9497831.

[13] S. Alazmi and D. C. de Leon, “A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerabil ity Scanners,” IEEE Access, vol. 10, pp. 33200–33219, 2022, doi: 10.1109/ACCESS.2022.3161522.

[14] “HAXSS: Hierarchical Reinforcement Learning for XSS Payload Gener ation,” IEEE publication, 2023. [Online]. Available: https://ieeexplore.i eee.org/document/10063589/

[15] M. Althunayyan, N. Saxena, S. Li, and P. Gope, “Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities,” Electronics, vol. 11, no. 13, Art. no. 2049, 2022, doi: 10.3390/electronics11132049.

[16] S. Calzavara, S. Roth, A. Rabitti, M. Backes, and B. Stock, “A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web,” in Proc. 29th USENIX Security Symposium, 2020, pp. 683 697.

[17] G. Karopoulos, D. Geneiatakis, and G. Kambourakis, “Neither Good nor Bad: A Large-Scale Empirical Analysis of HTTP Security Response Headers,” in Trust, Privacy and Security in Digital Business, Springer, 2021, pp. 83–95, doi: 10.1007/978-3-030-86586-3 6.

[18] U. Kishnani and S. Das, “Securing the Web: Analysis of HTTP Security Headers in Popular Global Websites,” arXiv:2410.14924, 2024. [Online]. Available: https://arxiv.org/abs/2410.14924

[19] S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, “Deep Learning Based Vulnerability Detection: Are We There Yet?” IEEE Transactions on Software Engineering, vol. 48, no. 9, pp. 3280–3296, 2022, doi: 10.1109/TSE.2021.3087402.

[20] "Combining Graph-based Learning with Automated Data Collection for Code Vulnerability Detection," 2021. [Online]. Available: https: //eprints.whiterose.ac.uk/id/eprint/168594/
[21] "SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities," IEEE Transactions on Dependable and Secure Comput ing, 2022, doi: 10.1109/TDSC.2021.3051525.
[22] A. Haddad, N. Aaraj, P. Nakov, and S. F. Mare, "Automated Map ping of CVE Vulnerability Records to MITRE CWE Weaknesses," arXiv:2304.11130, 2023. [Online]. Available: https://arxiv.org/abs/ 2304.11130
[23] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen, "Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses," IEEE Transactions on Dependable and Secure Computing, 2022, doi: 10.1109/TDSC.2021.3125270.
[24] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Exploit Prediction Scoring System (EPSS)," Digital Threats: Research and Practice, vol. 2, no. 3, Art. no. 20, 2021, doi: 10.1145/3436242.
[25] T. H. M. Le, H. Chen, and M. A. Babar, "A Survey on Data-driven Software Vulnerability Assessment and Prioritization," ACM Computing Surveys, vol. 55, no. 5, Art. no. 100, 2022, doi: 10.1145/3529757.
[26] X. Duan, M. Ge, T. H. M. Le, F. Ullah, S. Gao, X. Lu, and M. A. Babar, "Automated Security Assessment for the Internet of Things," arXiv:2109.04029, 2021. [Online]. Available: https://arxiv.org/abs/2109 .04029
[27] G. P. Bhandari, A. Naseer, and L. Moonen, "CVEfixes: Automated Collection of Vulnerabilities and Their Fixes from Open-Source Software," in Proc. 17th Int. Conf. on Predictive Models and Data Analytics in Software Engineering, 2021, doi: 10.1145/3475960.3475985.
[28] J. Mayer, M. Schramm, L. Bechtel, N. Lohmiller, S. Kaniewski, M. Menth, and T. Heer, "I Know Who You Scanned Last Summer: Mapping the Land scape of Internet-Wide Scanners," in Proc. IFIP Networking Conference, 2024, pp. 222–230, doi: 10.23919/IFIPNetworking62109.2024.10619808.
[29] J. Bau, E. Bursztein, D. Gupta, and J. C. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing," 2010.
[30] R. Aranovich et al., "Beyond NVD: Cybersecurity Meets the Semantic Web," in Proc. New Security Paradigms Workshop, 2021.
[31] O. Grigorescu, A. Nica, M. Dascalu, and R. Rughinis, "CVE2ATT&CK: BERT-Based Mapping of CVEs to MITRE ATT&CK Techniques," Algorithms, vol. 15, no. 9, Art. no. 314, 2022, doi: 10.3390/a15090314.
[32] "Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities," in Proc. 28th USENIX Security Symposium, 2019.
[33] National Institute of Standards and Technology, "National Vulnerability Database," 2024. [Online]. Available: https://nvd.nist.gov/
[34] National Institute of Standards and Technology, Guide to Enterprise Patch Management Planning, NIST SP 800-40 Rev. 4, 2022.
[35] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, Technical Guide to Information Security Testing and Assessment, NIST SP 800-115, 2008.
[36] OWASPFoundation, "Web Security Testing Guide," continuously updated. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/latest/
[37] FIRST, "Common Vulnerability Scoring System Version 4.0," 2023. [Online]. Available: https://www.first.org/cvss/v4-0/
[38] Cybersecurity and Infrastructure Security Agency, "Known Exploited Vulnerabilities Catalog," continuously updated. [Online]. Available: https: //www.cisa.gov/known-exploited-vulnerabilities-catalog
[39] ProjectDiscovery, "Nuclei: Fast and Customizable Vulnerability Scan ner Based on Simple YAML-Based Templates," 2023–2025. [Online]. Available: https://github.com/projectdiscovery/nuclei
[40] "Software Supply-Chain Security of Containerized IoT Components: A Comparative Vulnerability Assessment Using Trivy and Grype," Energies, MDPI, 2026, doi: 10.3390/en-trivy-grype.
[41] Google, "OSV: Open Source Vulnerabilities Database," continuously updated. [Online]. Available: https://osv.dev/
[42] National Institute of Standards and Technology, "NVD Program Tran sition to Risk-Based Enrichment," 2026. [Online]. Available: https: //nvd.nist.gov/general/news/nvd-program-transition-2
[43] J. Spring, E. Hatleback, A. Householder, A. Manion, and D. Shick, "Prioritizing Vulnerability Response: A Stakeholder-Specific Vulnerability Categorization (SSVC)," Software Engineering Institute, Carnegie Mellon University, Tech. Rep., 2021, updated 2024.
[44] FIRST EPSS Special Interest Group, "Exploit Prediction Scoring System: Model Updates v3–v5," 2023–2026. [Online]. Available: https://www.fir st.org/epss/
[45] OWASP CycloneDX and Linux Foundation SPDX, "Software Bill of Materials Standards: CycloneDX 1.7 and SPDX 3.0," 2024–2025. [Online]. Available: https://cyclonedx.org/ and https://spdx.dev/
[46] "Revealing the Black Box of Device Search Engines: Scanning Assets, Strategies, and Ethical Considerations," arXiv preprint, 2024