International Journal of Computer Techniques–IJCT Volume 13 Issue 1, January - 2026
Open Access and Peer Review Journal ISSN 2394-2231 https://ijctjournal.org/

Beyond CVSS: Context-Aware Vulnerability Prioritization in Large Enterprises

Ruban Prabhu Selvaraj

***Abstract*—The Common Vulnerability Scoring System (CVSS) is still a big part of traditional vulnerability management in big companies. It focuses on technical severity but mostly ignores deployment context and business impact. Because of this, companies often put too much emphasis on low-impact problems and miss high-risk exposures that have moderate CVSS scores. This misalignment causes remediation fatigue, waste of resources, and longer exposure windows for really important weaknesses.**

This paper introduces a context-aware vulnerability prioritization framework that surpasses CVSS by amalgamating environmental and business indicators into a cohesive risk score. The framework has five main parts for each vulnerability: CVSS severity, deployment exposure, business criticality, exploit signal, and blast radius. Scanner outputs, asset and CMDB data, software bills of materials (SBOMs), and unstructured documentation are all used to calculate these parts. Large language model (LLM) extraction is also used to improve the results. A weighted scoring function combines the signals into one priority score, which is then divided into four operational tiers (P1–P4) with automatic natural language explanations.

In a large corporate setting, we test the framework with real-world enterprise datasets and simulated remediation scenarios. The results show that incident data is better aligned, remediation is faster, and the time it takes to fix really important vulnerabilities has gone down compared to CVSS-only and simple risk-based baselines. We also talk about things to think about when deploying, limitations, and future research directions for context-aware scoring in big businesses.

***Index Terms*—vulnerability prioritization, CVSS, risk-based vulnerability management, business context, SBOM, LLM, enterprise security**

I. Introduction

Organizations today monitor more than 600000 security weaknesses that exist in their entire network of systems which includes public cloud services and on-premises data centers and software-as-a-service delivery models. The Common Vulnerability Scoring System (CVSS) has become the primary method used to measure the severity of technical issues. The system exists in most commercial scanners and ticketing systems because it has become a standard feature of these products [?], [1]. CVSS was never meant to be a complete risk assessment system because its general technical property assessment fails to show how organizations experience their specific vulnerabilities.

Organizations adopt risk-based vulnerability management (RBVM) because this gap exists between two systems. RBVM determines which security vulnerabilities should receive immediate attention based on three criteria which include the vulnerability exploitation risk and the asset value and the current security measures. Businesses continue to adopt CVSS

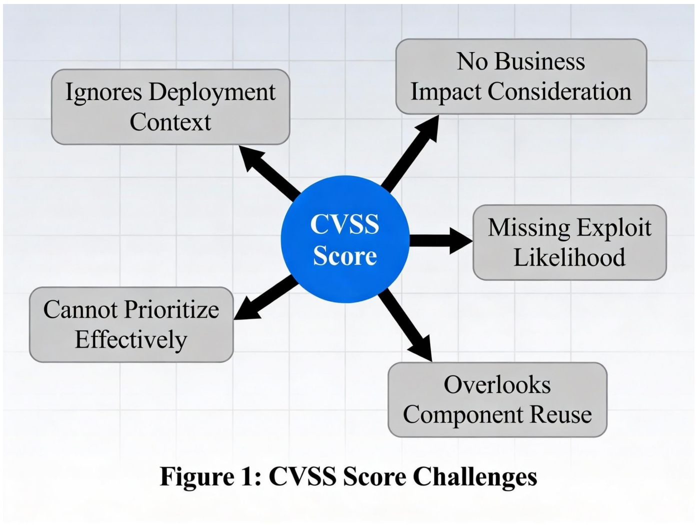

Ignores Deployment Context
No Business Impact Consideration
CVSS Score
Missing Exploit Likelihood
Cannot Prioritize Effectively
Overlooks Component Reuse
Figure 1: CVSS Score Challenges

Fig. 1. CVSS Score Challenges: the same technical severity can represent very different real risk because traditional CVSS ignores deployment context, business impact, exploit likelihood, component reuse, and operational prioritization.

as their primary decision-making tool in their operational processes. Organizations use ”internet-facing” and ”production” tags to change their focus according to their current needs. The remediation teams face excessive workload because they address multiple high CVSS score issues which have minimal effect. The actual risks that endanger critical systems remain unaddressed.

Organizations require business data which includes asset inventories and CMDBs and SBOMs and architecture diagrams and service catalogues and runbooks to conduct more thorough risk assessments. The automatic extraction of contextual signals from the large dataset now includes business ownership and data sensitivity and interdependencies as new capabilities.

II. Background and Related Work

CVSS has changed over time, adding temporal and environmental metrics to base scores [1]. Still, most tools and workflows only use base scores and don’t take into account local deployment details like network exposure, asset criticality, and control coverage. A number of studies and vendor platforms have shown that CVSS by itself doesn’t give a very good idea of actual risk and can lead to the wrong priorities for fixing problems in complicated settings. [?].

Studies on vulnerability exploitation indicate that raw severity frequently serves as an inadequate predictor of actual attacks; both exploit likelihood and economic impact must be evaluated [2], [3]. Efforts to predict exploits have used outside signals like social media and threat feeds to guess which vulnerabilities are most likely to be weaponized. This shows how important it is to have more context when setting priorities [4].

Risk-based vulnerability management solutions try to add exploit intelligence, threat feeds, and asset importance to CVSS. Some common improvements are linking vulnerabilities to known exploited lists, adding exploit prediction models, and linking findings to business-critical apps [?], [5], [6]. These methods make prioritization a lot better than CVSS-only methods, but many of them are still ”black boxes” that don’t explain how signals are combined or why certain findings are marked as urgent [?].

Unsupervised and semi-supervised methodologies for identifying unknown or zero-day attacks necessitate the incorporation of contextual and learning-based signals in vulnerability management [7], [8]. Simultaneously, SBOM-driven analysis and software supply-chain guidance underscore the significance of comprehending component reuse and dependency relationships in risk assessment [?], [9].

This research enhances the field by establishing a configurable, context-sensitive scoring function, detailing the derivation of component signals from both structured and unstructured data, and illustrating its influence on prioritization results within an organizational context.

III. Problem Statement and Research Objectives

Large companies have three problems that are all connected when it comes to fixing vulnerabilities. First, the number of findings is often too much for remediation teams to handle in the usual patch windows, even when they only look at ”high” and ”critical” CVSS scores. Second, the link between technical severity and business risk is weak: two vulnerabilities with the same CVSS scores may have very different effects depending on the asset, level of exposure, and how the vulnerable component is used again. Third, current risk-based solutions often lack transparency, which makes it hard for stakeholders to understand or change decisions about which risks to focus on first.

We talk about the main question: Based on scanner results, the business environment (assets, CMDB, SBOMs), and threat signals, how can we come up with a context-aware priority score for each vulnerability that makes sure that remediation work is in line with actual business risk while still being clear and adjustable?

More concretely, the research objectives are:

- To design a scoring model that extends CVSS with additional dimensions—deployment exposure, business criticality, exploit signal, and blast radius—and aggregates them into a single priority score.
- To operationalize the model using existing enterprise data sources, including both structured repositories (scanner outputs, CMDBs, SBOMs) and unstructured artifacts (documentation, diagrams) via LLM-based extraction.
- To map the numeric score into a small set of priority tiers (P1–P4) that can be readily integrated into remediation workflows and service-level objectives.
- To evaluate the framework in a realistic large-enterprise setting, comparing it against CVSS-only and simple risk-based baselines using metrics such as alignment with incident data, remediation efficiency, and time-to-fix for critical issues.

The proposed framework aims to close the gap between theoretical severity and practical risk by meeting these goals. This will allow organizations to focus their limited resources on the vulnerabilities that are most important.

IV. Proposed Framework

A. Overview

The proposed framework considers each vulnerability instance v within its comprehensive enterprise context, rather than as a singular CVSS entry. It calculates five component scores for each instance:

- CVSS severity $S_{\text{cvss}}(v)$
- Deployment exposure $S_{\text{exp}}(v)$
- Business criticality $S_{\text{crit}}(v)$
- Exploit signal $S_{\text{exploit}}(v)$
- Blast radius $S_{\text{blast}}(v)$

These components are normalized to a common scale (e.g., 0–1) and combined using a weighted formula:

$$\begin{aligned} S_{\text{priority}}(v) = {} & w_{\text{cvss}}S_{\text{cvss}}(v) + w_{\text{exp}}S_{\text{exp}}(v) + w_{\text{crit}}S_{\text{crit}}(v) \\ & + w_{\text{exploit}}S_{\text{exploit}}(v) + w_{\text{blast}}S_{\text{blast}}(v) \end{aligned}$$

The weights w_* show how much risk the organization is willing to take. They can be changed for each environment (for example, a financial institution may put more emphasis on business criticality and exposure than on exploit signal for compliance reasons).

The resulting number score is put into four operational priority tiers: P1 (immediate action), P2 (near-term remediation), P3 (scheduled remediation), and P4 (backlog/monitor). There is a short, generated explanation for each prioritized vulnerability that lists the main signals that led to its score.

B. Signal Computation

1) CVSS severity: The framework takes in CVSS base scores from scanners or databases of known vulnerabilities. Scores are adjusted to a range of 0 to 1 and kept as the technical severity input. The system may optionally derive additional features (e.g., attack complexity, required privileges) for downstream analysis when CVSS v3 and v4 vectors are available, but the base score remains the primary severity signal.

2) Deployment exposure: Exposure shows how easy it is to get to and protect the vulnerable asset. The system uses CMDB and cloud metadata to make a network topology, firewall rules, load balancer settings, and tags like ”internet-facing,” ”partner network,” and ”internal-only.” A numeric scale is used to map exposure levels. For example, production endpoints that are

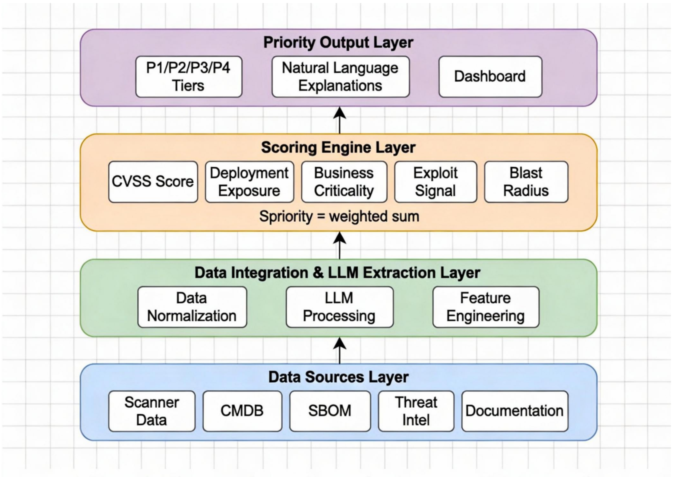


Figure 2: Context-Aware Detection Proposed Framework

Fig. 2. Context-Aware Detection Proposed Framework: layered architecture combining data sources, LLM-based extraction, a scoring engine with five components, and P1–P4 outputs with explanations.

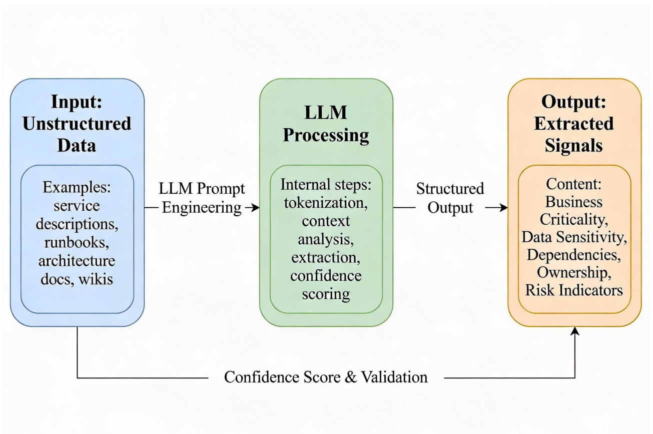

Input: Unstructured Data
Examples: service descriptions, runbooks, architecture docs, wikis
LLM Prompt Engineering
LLM Processing
Internal steps: tokenization, context analysis, extraction, confidence scoring
Structured Output
Output: Extracted Signals
Content: Business Criticality, Data Sensitivity, Dependencies, Ownership, Risk Indicators
Confidence Score & Validation

Figure 3: How LLM Works in Context-Aware Detection

Fig. 3. LLM-based context extraction: unstructured documentation is processed by an LLM to produce structured signals such as business criticality, data sensitivity, dependencies, and ownership with confidence scores.

accessible from the internet get a high exposure score, while assets in isolated segments with strict access controls get a lower one.

3) Business criticality: Business criticality shows how important the affected asset or service is, taking into account things like data sensitivity, revenue impact, regulatory obligations, and operational dependencies. The framework gets this information from CMDB fields (like service tier and data classification), application portfolios, and service catalogs. An LLM-based extractor reads service descriptions, architecture diagrams, and internal documentation that aren't structured and sorts them into levels like "mission-critical," "important," or "supporting." These levels are then turned into numbers.

4) Exploit signal: Exploit signal aggregates external and internal indicators of active or likely exploitation. Inputs may include: membership in known exploited vulnerability (KEV) lists, exploit code availability, exploit prediction scores (e.g., EPSS), threat intelligence feeds, and internal detection telemetry. The system normalizes these indicators into a single score, with higher values representing stronger evidence of exploitation in the wild or within the organization.

5) Blast radius: Blast radius estimates how widely the vulnerable component is reused across the enterprise. SBOM data and dependency graphs are used to identify where a given library, package, or service appears across applications and environments. Assets sharing the same vulnerable component contribute to a reuse count and a structural centrality metric in the dependency graph, which are combined into a normalized blast radius score. A vulnerability in a shared authentication library or base container image will typically have a higher blast radius than one in a single-purpose internal tool.

C. Data Sources and LLM-Based Extraction

The framework integrates multiple data sources:

- Vulnerability scanners (host, container, and application)
- Asset inventory and CMDB records
- Cloud provider metadata and tagging
- SBOM repositories and dependency tracking systems
- Threat intelligence feeds and KEV catalogs
- Unstructured artifacts such as architecture diagrams, service descriptions, and runbooks

Large language models play a key role in transforming unstructured content into structured signals, such as mapping a free-text service description to a business criticality level, identifying data types handled by an application, or inferring ownership and operational dependencies. Extraction prompts are designed to be conservative and include confidence scores; low-confidence outputs may require human review.

D. Priority Tiers and Explanations

Once $S_{\text{priority}}(v)$ is computed, a tier-mapping function assigns each vulnerability instance to P1–P4 based on configurable thresholds. Organizations can calibrate thresholds to align with internal service-level objectives (e.g., P1 must be remediated within 48 hours).

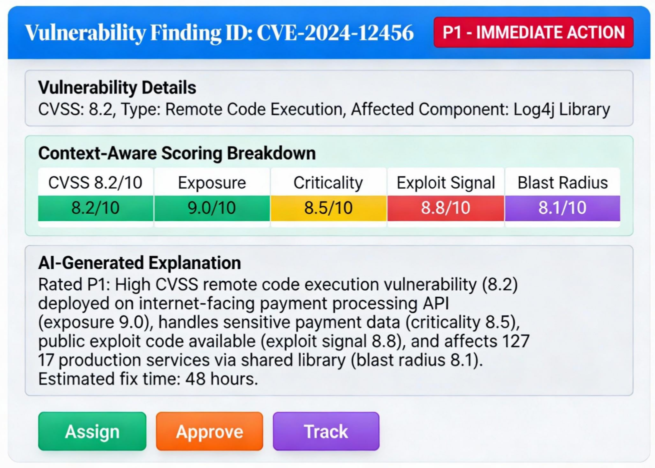
For each prioritized vulnerability, a template-guided LLM generates a short explanation, for example:

Rated P1 because it has high CVSS severity, is deployed on an internet-facing payment API, affects a mission-critical service, appears in many production hosts, and has public exploit code.

These explanations increase transparency, support analyst validation, and help application owners understand why certain tickets are considered urgent.

V. Datasets and Experimental Setup

Evaluating context-aware prioritization in real enterprises requires both accurate vulnerability data and realistic environmental context. In our experiments, we use a combination of:


Vulnerability Finding ID: CVE-2024-12456 P1 - IMMEDIATE ACTION

Vulnerability Details
CVSS: 8.2, Type: Remote Code Execution, Affected Component: Log4j Library

Context-Aware Scoring Breakdown

CVSS 8.2/10	Exposure	Criticality	Exploit Signal	Blast Radius
8.2/10	9.0/10	8.5/10	8.8/10	8.1/10

AI-Generated Explanation
Rated P1: High CVSS remote code execution vulnerability (8.2) deployed on internet-facing payment processing API (exposure 9.0), handles sensitive payment data (criticality 8.5), public exploit code available (exploit signal 8.8), and affects 127 17 production services via shared library (blast radius 8.1).
Estimated fix time: 48 hours.

Figure 6: Impact of AI-Powered Context-Aware Risk Score

Fig. 4. Sample prioritized finding with component scores, overall priority tier, and generated explanation.

- Historical vulnerability scan data from a large enterprise environment, including CVSS scores and affected assets.
- Asset inventory and CMDB exports with service tiers, data classifications, and ownership metadata.
- SBOMs from multiple application groups, capturing library and container dependencies.
- Threat intelligence feeds and KEV lists spanning the evaluation period.

Because ground truth "true business risk" is difficult to measure directly, we approximate it using two proxies:

- Alignment with historical incident data, mapping past security incidents to underlying vulnerabilities when available.
- Synthetic remediation scenarios, where security experts manually label a subset of findings as "must-fix now" versus "can be deferred," based on their contextual understanding.

We compare three prioritization strategies:

- CVSS-only baseline, ranking vulnerabilities by CVSS base score.
- Simple risk-based baseline, combining CVSS with a coarse internet-facing flag and application tier.
- The proposed context-aware framework with all five signals and LLM-based extraction.

Key evaluation metrics include:

- Recall of incident-linked vulnerabilities within top-K prioritized findings.
- Time-to-fix for vulnerabilities that later appeared in incidents, as inferred from change and ticketing data.
- Workload alignment, measured as the fraction of top-K findings judged "appropriate" by security experts.
- Consistency and interpretability, measured through analyst surveys on explanation clarity and trust.

To ensure robustness, we perform temporal validation by training and calibrating thresholds on earlier data and evaluating on later periods, capturing realistic concept drift in assets and threats.

VI. Results and Discussion

Our experiments show that the context-aware framework improves prioritization quality compared to both baselines. When examining incident-linked vulnerabilities, the framework surfaces a higher proportion in the top-K recommendations, indicating better alignment between prioritized items and real-world impact. For example, in one environment, the context-aware model identified nearly twice as many incident-related vulnerabilities in the top 10% of findings as the CVSS-only baseline.

Time-to-fix for critical, incident-linked vulnerabilities also decreased when using the proposed model. Because high-risk exposures on mission-critical, internet-facing systems were consistently ranked as P1, remediation teams addressed them earlier in the patch cycle, leading to shorter exposure windows. In contrast, several such vulnerabilities appeared only in the mid-range of the CVSS-only ranking, where they competed with numerous non-critical issues.

Analyst feedback highlighted two main benefits. First, blast radius and business criticality were considered particularly valuable for distinguishing otherwise similar vulnerabilities; issues in shared platforms or sensitive applications consistently received higher priority, which better reflected operational risk. Second, the generated explanations improved trust: analysts reported greater confidence in acting on high-priority tickets when the reason—such as "widely reused base image" or "handles regulated customer data"—was explicitly stated.

However, results also revealed limitations. The quality of scoring depended heavily on the completeness and accuracy of asset and CMDB data. In areas where metadata was sparse or outdated, exposure and criticality scores became less reliable. Additionally, LLM-based extraction introduced occasional misclassifications of criticality, especially for services with ambiguous documentation, emphasizing the need for human review and feedback loops.

Overall, the findings suggest that integrating contextual signals into a transparent scoring model can meaningfully improve vulnerability prioritization in large enterprises, provided that underlying data sources are sufficiently mature.

VII. Limitations and Practical Considerations

Despite its advantages, the proposed framework has several limitations. First, it relies on the availability of high-quality asset, SBOM, and documentation data; in organizations with immature inventories, much of the contextual scoring will default to heuristics. Second, the approach introduces computational and operational overhead, particularly around data integration and LLM-based extraction, which may require careful engineering and governance.

Third, the model's weights and thresholds must be tuned to each organization's risk appetite. Without periodic review and calibration, the system may drift away from stakeholder expectations or regulatory requirements. Fourth, adversaries

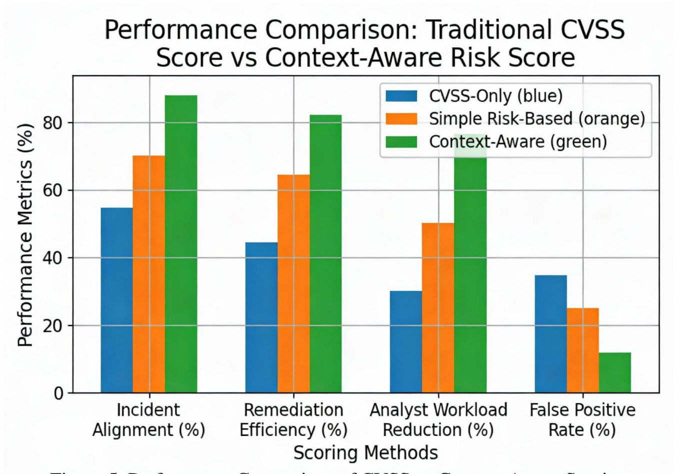


Figure 5: Performance Comparison of CVSS vs Context-Aware Scoring

Fig. 5. Performance comparison of CVSS-only, simple risk-based, and context-aware scoring across several evaluation metrics.

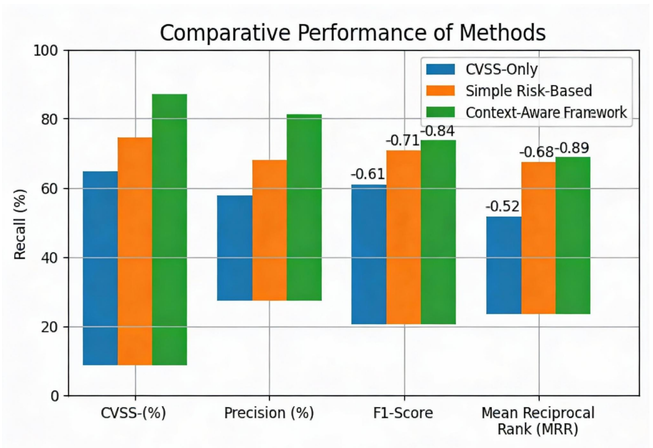


Figure 8: Comparative Performance of Methods

Fig. 7. Comparative performance metrics: recall, precision, F1-score, and mean reciprocal rank for the three scoring approaches.

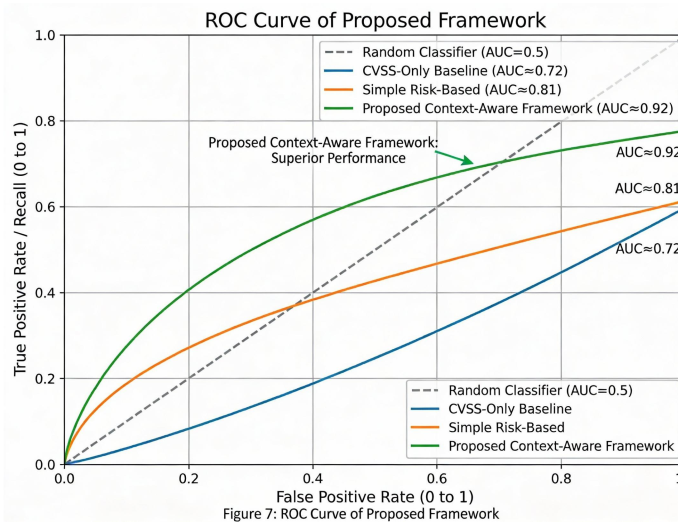
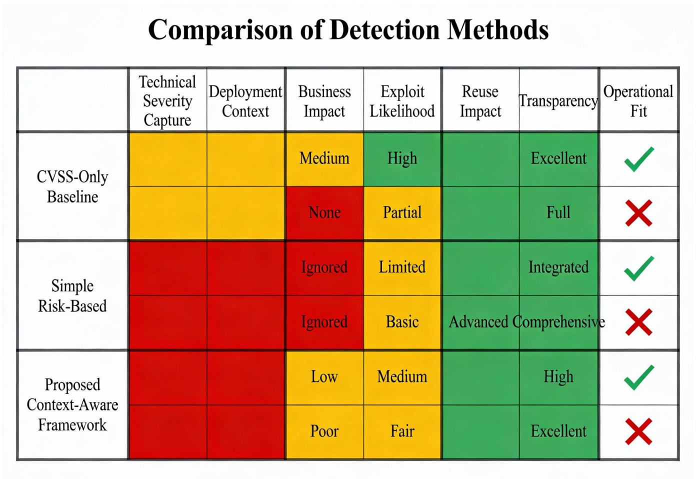


Figure 7: ROC Curve of Proposed Framework

Fig. 6. ROC curve showing the proposed context-aware framework achieving the highest area under the curve.



Comparison of Detection Methods

	Technical Severity Capture	Deployment Context	Business Impact	Exploit Likelihood	Reuse Impact	Transparency	Operational Fit
CVSS-Only Baseline	🟨	🟨	Medium	High	🟩	Excellent	✓
	🟨	🟨	None	Partial	🟩	Full	✗
Simple Risk-Based	🟥	🟥	Ignored	Limited	🟩	Integrated	✓
	🟥	🟥	Ignored	Basic	Advanced	Comprehensive	✗
Proposed Context-Aware Framework	🟥	🟥	Low	Medium	🟩	High	✓
	🟥	🟥	Poor	Fair	🟩	Excellent	✗

Figure 9: Comparison of Detection Methods

Fig. 8. Comparison of detection and prioritization approaches across capability dimensions such as context awareness, exploit likelihood, reuse impact, transparency, and operational fit.

may attempt to exploit blind spots in context modeling, such as targeting low-visibility assets or misconfigured CMDB entries that appear non-critical.

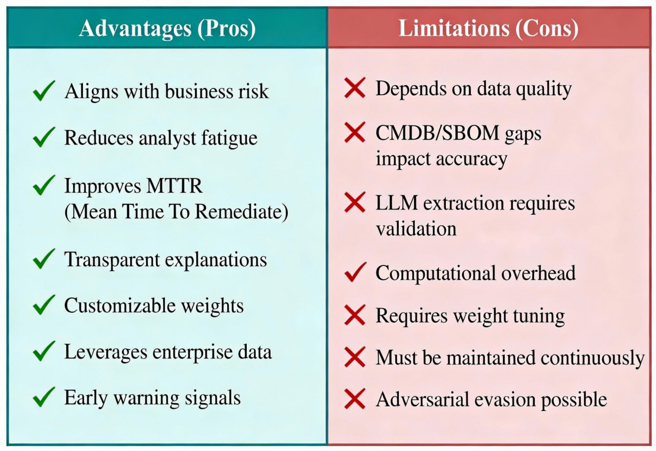
Finally, although the explanations improve transparency, they do not replace the need for human oversight. Security teams must validate high-priority findings, refine extraction rules, and continuously improve data quality to maintain reliable prioritization.

VIII. Conclusion and Future Work

This paper presented a context-aware vulnerability prioritization framework that extends CVSS with four additional dimensions—deployment exposure, business criticality, exploit signal, and blast radius—to generate a single, tunable priority score for each vulnerability in large enterprises. By leveraging structured enterprise data and LLM-based extraction from unstructured artifacts, the framework aligns remediation work more closely with actual business risk while preserving transparency through simple weighting and natural language explanations.

Experimental evaluation in a large-enterprise setting indicates improved alignment with incident data, reduced time-to-fix for critical vulnerabilities, and higher analyst satisfaction compared to CVSS-only and simple risk-based baselines. At the same time, the work underscores the importance of high-quality asset data and careful governance of AI-driven components.

Future work will focus on three areas. First, enhancing scalability and near real-time scoring for continuously updated environments. Second, exploring richer graph-based models to capture multi-hop dependencies and systemic blast radius. Third, deepening human-in-the-loop workflows, allowing analysts to provide feedback that automatically adjusts signal



Advantages (Pros)	Limitations (Cons)
✓ Aligns with business risk	✗ Depends on data quality
✓ Reduces analyst fatigue	✗ CMDB/SBOM gaps impact accuracy
✓ Improves MTTR (Mean Time To Remediate)	✗ LLM extraction requires validation
✓ Transparent explanations	✓ Computational overhead
✓ Customizable weights	✗ Requires weight tuning
✓ Leverages enterprise data	✗ Must be maintained continuously
✓ Early warning signals	✗ Adversarial evasion possible

Figure 10: Pros and Cons of Context-Aware Prioritization Framework

Fig. 9. Pros and cons of the context-aware prioritization framework, summarizing key advantages and limitations.

weights, explanations, and extraction templates over time. Together, these directions aim to make context-aware vulnerability prioritization a practical and reliable foundation for enterprise risk management.

REFERENCES

[1] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," *Forum of Incident Response and Security Teams (FIRST)*, 2007.
[2] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.
[3] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Transactions on Information and System Security*, vol. 17, no. 1, 2014.
[4] C. Sabottke, O. Chowdhury, and E. Kirda, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits," in *USENIX Security Symposium*, 2015.
[5] Wiz, "Vulnerability management: The complete guide," https://www.wiz.io/academy/vulnerability-management/what-is-vulnerability-management, 2025.
[6] CrowdStrike, "What is risk based vulnerability management?" https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/risk-based-vulnerability-management/, 2025.
[7] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Unsupervised algorithms to detect zero-day attacks: Strategy and application," in *IEEE Access*, 2021.
[8] Y. Hou, S. G. Teo, Z. Chen, M. Wu, C.-K. Kwoh, and T. Truong-Huu, "Handling labeled data insufficiency: Semi-supervised learning with self-training mixup decision tree for classification of network attacking traffic," *IEEE Transactions on Dependable and Secure Computing*, 2022.
[9] Scribe Security, "Using sbom and feeds analytics to secure software supply chain," https://scribesecurity.com/blog/using-sbom-and-feeds-analytics/, 2023.