

AI Driven Mission Aware Vulnerability Prioritization for Cloud Native Critical Infrastructure

Md. Humayun Kabir¹, Md. Golam Mostafa², Shakila Akter³

¹Master of Science in Information Technology Westcliff University, United State

²National University, Ensure Support Services Limited, Bangladesh

³Master of Science in Computer and Information Science, Lewis University, United States



DOI : <https://doi.org/10.61796/ipteks.v3i4.626>



Sections Info

Article history:

Submitted: June 20, 2026

Final Revised: July 15, 2026

Accepted: September 04, 2026

Published: September 30 2026

Keywords:

Vulnerability prioritization

Cloud-native security

Critical infrastructure

Mission assurance

Explainable artificial intelligence

Dependency graphs

Cascading impact

Remediation governance

ABSTRACT

Objective: Cloud-native critical infrastructure concentrates operational responsibility in elastic, highly connected systems whose software composition and runtime topology change continuously. Organizations therefore face a prioritization problem rather than a vulnerability-discovery problem: thousands of known vulnerabilities may be present, yet only a subset is reachable, exploitable, attached to mission-essential assets, or capable of producing cascading service disruption. The Common Vulnerability Scoring System (CVSS) remains valuable for describing intrinsic technical severity, but a severity-first queue can misallocate scarce remediation capacity. **Method:** This article proposes the Mission-Aware Vulnerability Prioritization Framework (MAVPF), an explainable artificial-intelligence architecture that unifies six evidence domains: runtime exposure, exploitability, asset criticality, critical-service dependency, mission impact, and cascading operational impact. MAVPF represents cloud resources, identities, software components, and services as a time-aware dependency graph; fuses scanner, runtime, threat-intelligence, service-catalog, and resilience evidence; produces a constrained remediation ranking; and generates a traceable explanation and recommended action for every vulnerability. Its theoretical foundation integrates risk-based vulnerability management, attack-graph reasoning, socio-technical resilience, graph-based dependency analysis, and explainable AI. **Results:** A worked scenario shows why an exposed, currently exploitable CVSS 7.5 vulnerability supporting an essential service can rationally outrank an unexposed CVSS 9.8 vulnerability on a low-criticality service. The article does not report empirical performance; instead, it specifies falsifiable evaluation questions, baselines, metrics, and validation procedures for future testbed and field studies. **Novelty:** The contribution is a generalizable decision architecture that connects technical vulnerability evidence to mission consequences without replacing CVSS or concealing uncertainty.

INTRODUCTION

Vulnerability management has become a problem of constrained choice. Modern organizations routinely identify more weaknesses than they can patch within a single maintenance cycle, while cloud-native environments continuously instantiate containers, functions, identities, APIs, and managed services. A remediation queue ordered primarily by CVSS can appear objective yet remain operationally misleading. CVSS describes the characteristics and potential severity of a vulnerability; its own specification distinguishes Base, Threat, and Environmental metric groups and permits consumers to incorporate contextual information [1]. That flexibility, however, does not itself establish which affected workload sustains a critical mission, whether the vulnerable component is live and reachable, or how failure could propagate across dependent services.

The practical distinction is visible in the motivating case. CVE A has a CVSS score of 9.8 but is not exposed, affects a low-criticality service, and has limited plausible operational impact. CVE B has a score of 7.5 but is reachable, has credible exploitation evidence, supports an essential service, and could impair downstream services. A severity-first policy selects CVE A. A mission-risk policy can select CVE B, provided that the decision is transparent, bounded by evidence, and sensitive to uncertainty. This is not an argument that severity is irrelevant. It is an argument that technical severity is one input to a different decision: which remediation action most reduces expected mission disruption under limited resources.

Empirical research supports moving beyond severity alone. Exploitation is highly uneven across disclosed vulnerabilities, and observed exploitation provides information not captured by static severity labels [2], [3], [4]. Predictive approaches such as the Exploit Prediction Scoring System (EPSS) estimate the probability of exploitation activity, while decision frameworks such as Stakeholder-Specific Vulnerability Categorization (SSVC) emphasize contextual decisions rather than universal numeric rankings [5], [6]. These advances narrow the remediation set but do not fully resolve the cloud-native mission question: what happens to essential services if a particular vulnerable component is compromised now?

This article addresses that narrower gap by proposing one explainable framework that integrates runtime exposure, exploitability, asset criticality, critical-service dependency, mission impact, and cascading operational impact. It focuses exclusively on known-vulnerability remediation. It is not an attack-detection system, a zero-day detector, an autonomous containment controller, or another universal vulnerability-severity scale. The principal research question is: How can AI prioritize known vulnerabilities according to their actual operational and mission risk to cloud-native critical infrastructure rather than primarily their technical severity?

The framework also aligns with U.S. cybersecurity policy priorities. Executive Order 14028 identifies the prevention, detection, assessment, and remediation of cyber incidents as essential to national and economic security and directs modernization toward secure cloud services, risk analytics, and stronger software-supply-chain practices [7]. CISA's Cross-Sector Cybersecurity Performance Goals emphasize risk-based vulnerability management for critical infrastructure, while the NIST Cybersecurity Framework 2.0 supports governance and prioritization based on organizational context and consequences [8], [9]. MAVPF operationalizes these priorities by linking known vulnerabilities in cloud-native systems to exposure, service dependencies, mission consequences, and auditable remediation decisions.

The article makes four contributions. First, it defines mission-aware vulnerability priority as a decision about marginal reduction in service risk, rather than as a restatement of technical severity. Second, it introduces a time-aware evidence graph connecting vulnerabilities to live software instances, assets, identities, services, mission functions, and cross-service dependencies. Third, it proposes a hybrid AI architecture in

which learned models infer uncertain relationships and likely consequences, while explicit policy constraints preserve governance, auditability, and human authority. Fourth, it specifies an evaluation design that can test ranking quality, mission-loss reduction, explanation fidelity, robustness to incomplete telemetry, and generalization across organizations and infrastructure sectors.

The framework is intentionally generalizable. It does not assume one cloud provider, one industry, or one proprietary business-impact taxonomy. Organizations can configure mission functions and action thresholds while retaining a common evidence schema and evaluation protocol. This separation between transferable architecture and local mission knowledge is essential for critical-infrastructure relevance.

CVSS supplies a standardized description of vulnerability characteristics and potential technical impact. Version 4.0 separates intrinsic Base metrics from time-varying Threat metrics and organization-specific Environmental metrics, improving the vocabulary available to consumers [1]. The system is indispensable for comparability, triage, and communication. Yet a Base score is not a measurement of an organization's operational loss, and the specification cautions consumers to use environmental information. Treating Base severity as a complete remediation policy therefore commits a category error: it converts a technical descriptor into a mission decision without modeling the affected system.

Research has repeatedly found that vulnerability disclosure and severity do not map uniformly to exploitation. Bozorgi et al. [3] demonstrated that machine learning could help distinguish vulnerabilities likely to be exploited. Allodi and Massacci [2] compared vulnerability information with evidence from attack data and showed that black-market and exploitation signals alter risk interpretation. Sabottke, Suciu, and Dumitraş [4] used social-media signals to forecast exploitation, while Suciu et al. [10] demonstrated improved prediction of exploit development across a large vulnerability corpus. These studies justify exploitability modeling but generally stop before tracing an exploited weakness to the loss of a specific critical service.

EPSS operationalizes one important component by estimating the probability that a published vulnerability will be exploited in the wild during a future time window [5]. SSVC reframes vulnerability management as stakeholder-specific decisions based on factors such as exploitation status, exposure, and mission impact [6]. The U.S. Cybersecurity and Infrastructure Security Agency's Known Exploited Vulnerabilities catalog similarly provides authoritative evidence of exploitation and supports time-bounded remediation for covered federal systems [11]. These mechanisms substantially improve prioritization over a CVSS-only queue.

However, exploitability and known exploitation do not alone express operational consequence. A vulnerable internet-facing development system and a vulnerable internal control-plane component may have different probabilities of exploitation, yet the latter could sustain a mission-essential service through privileged dependencies. Conversely, a severe and exploitable package may be present in an image but never loaded, invoked,

reachable, or deployed. Contextual prioritization therefore requires both executable-path evidence and service-dependency evidence.

Cloud-native systems complicate asset identity. Containers are ephemeral, software bills of materials describe composition rather than execution, and logical services span clusters, regions, identities, queues, databases, and external providers. A scan finding may refer to a dormant image layer, a running container, an admission-blocked workload, or a reachable service instance. Vulnerability Exchange (VEX) documents can communicate whether a product is affected, but runtime telemetry is needed to determine whether vulnerable code is loaded, reachable, and protected by compensating controls. NIST guidance on enterprise patch management consequently treats prioritization as a risk-response process rather than a mechanical score sort [12].

Runtime exposure should be understood as a structured claim supported by observations: deployment state, network reachability, invocation or code-path evidence, identity permissions, protection coverage, and telemetry freshness. Each claim has provenance and confidence. This prevents a binary exposed/not-exposed flag from hiding uncertainty or temporal drift.

Attack graphs model how adversaries may traverse prerequisites and privileges to reach security goals [13], [14]. Interdependency research in critical infrastructure shows that localized disruption can propagate through physical, cyber, geographic, and logical dependencies [15]. Service-resilience engineering likewise emphasizes graceful degradation, redundancy, recovery objectives, and common-mode failure. These traditions provide the theoretical bridge from compromised component to operational consequence.

A cloud-native dependency graph differs from a static network attack graph in two respects. First, it must represent service semantics: which workloads implement which service capabilities, what redundancy exists, and what minimum capacity is required. Second, it must be temporal: autoscaling, failover, configuration drift, and deployments alter both attack paths and mission dependencies. Mission impact is therefore not identical to asset value. An ordinary-looking identity, secret, message broker, or service mesh component may become critical because many essential services depend on it.

Security prioritization is a high-consequence decision-support task. Explanations must do more than display feature importance. They should identify the evidence path from vulnerability to mission consequence, distinguish observation from inference, disclose missing or stale data, show counterfactual conditions that would change priority, and record the governing policy. Explainable-AI research warns that post-hoc explanations can be unstable or unfaithful to the underlying model [16]–[18]. For this reason, MAVPF combines interpretable rules and graph paths with constrained learned components, rather than relying on an unconstrained opaque ranker.

Table 1. What established approaches contribute and what remains unresolved

Approach	Primary contribution	Unresolved mission-aware issue
CVSS	Standardized technical severity and environmental vocabulary	Does not establish live reachability, service dependency, or cascading mission loss by itself
EPSS and exploit prediction	Likelihood-oriented evidence for exploitation activity	Probability of exploitation is not the same as consequence to a critical service
KEV and threat intelligence	Evidence that exploitation is occurring or has occurred	Catalog status does not encode an organization's runtime topology or mission dependencies
SSVC	Context-sensitive decision logic for stakeholder actions	Requires reliable, current inputs and does not prescribe a cloud-native dependency model
Attack graphs	Paths from attacker prerequisites to security goals	Often asset- or privilege-centric rather than service-capability and resilience-centric
Cloud scanners and SBOMs	Broad discovery of components and known CVEs	Presence does not prove execution, reachability, or operational consequence
Business impact analysis	Mission importance, recovery needs, and tolerances	Often periodic and disconnected from rapidly changing technical telemetry

The literature offers strong components but weak integration. Severity systems describe vulnerability characteristics; exploit-prediction systems estimate attacker activity; exposure tools identify reachable assets; service catalogs document ownership; attack graphs model paths; and business-impact analyses describe essential functions. The missing unit of analysis is the complete, explainable evidence chain linking a known CVE to a currently running and reachable component, to the asset and service it supports, to mission degradation, and to plausible cascading consequences. Existing tools commonly aggregate these factors into vendor-specific risk scores without exposing model assumptions, uncertainty, or dependency paths. This limits scientific reproducibility and operational trust.

RESEARCH METHOD

This article adopts a design-science methodology: identify the decision problem, derive requirements from prior research and operational constraints, construct an artifact, demonstrate its logic with a transparent scenario, and specify a reproducible empirical evaluation protocol [19]. The present contribution remains conceptual and architectural until the proof-of-concept protocol described in Section 4.1 is executed and its run artifacts are independently reviewable. Accordingly, this version does not claim measured superiority over severity-first or exploitability-first baselines.

The framework's unit of decision is a vulnerability-instance pair, not merely a CVE. One CVE may appear in several images or workloads with different reachability, privileges, compensating controls, service dependencies, and mission consequences. The system may aggregate instances for campaign planning, but it preserves instance-level evidence so that a dormant library does not inherit the priority of an exposed production workload – or vice versa.



Continuous evidence refresh, uncertainty checks, analyst feedback, and audit logging

Figure 1. Mission Aware Vulnerability Prioritization Framework data and decision flow

MAVPF maintains a typed, time-aware evidence graph. Nodes represent CVEs, packages, images, workloads, hosts, identities, network endpoints, data stores, technical services, critical-service capabilities, mission functions, resilience controls, and remediation actions. Edges represent relations such as contains, runs, exposes, can-reach, authenticates-as, depends-on, provides-capability-to, fails-over-to, and impacts. Every node and edge carries provenance, observation time, confidence, ownership, and applicable policy scope.

Table 2. Integrated evidence domains

Domain	Representative evidence	Decision meaning	Key uncertainty
Technical severity	CVSS vector, affected version, vulnerability class	Potential technical harm if exploited	Vendor scoring quality and environmental fit

Domain	Representative evidence	Decision meaning	Key uncertainty
Exploitability	EPSS, KEV, exploit maturity, active campaigns, prerequisite complexity	Credibility and immediacy of exploitation	Threat intelligence lag and attacker adaptation
Runtime exposure	Deployment state, reachability, code-path use, ingress, identity permissions	Whether exploitation is feasible in the present environment	Ephemerality, telemetry gaps, stale topology
Asset criticality	Ownership, data sensitivity, privilege, recovery objectives	Importance of the directly affected resource	Inventory quality and inconsistent labels
Service dependency	Service map, traces, mesh data, configuration, runbooks	Which critical capability relies on the resource	Hidden dependencies and third-party services
Mission impact	Service tier, safety relevance, essential-function tolerance, affected population	Magnitude and urgency of operational harm	Subjective valuations and changing mission state
Cascade impact	Shared dependencies, blast paths, capacity thresholds, failover constraints	Secondary and correlated service degradation	Nonlinear behavior and common-mode failures

Stage one validates the finding. The framework reconciles scanner output with package inventories and deployment metadata, suppresses duplicates, and records whether the vulnerable component is present in a live instance. Stage two evaluates exploitability using multiple signals rather than a single feed. Stage three evaluates runtime feasibility: ingress and lateral reachability, exposed interfaces, authentication requirements, privileges, loaded code paths, segmentation, and compensating controls. Stage four resolves the asset-to-service-to-mission chain. Stage five estimates direct and cascading operational consequences under current capacity and failover conditions. Stage six ranks remediations, produces explanations, and recommends actions subject to organizational constraints.

The system should use different AI techniques for different evidence problems. Entity resolution can reconcile inconsistent identifiers across scanners, registries, clusters, and service catalogs. Graph learning can propose missing dependency edges and identify structurally consequential shared components. Calibrated exploit models can update likelihood evidence. Scenario models can estimate service degradation under component loss. A learning-to-rank component can order vulnerability-instance pairs from historical outcomes and expert preferences. None of these components should be allowed to silently override mandatory policies, such as deadlines for known exploited vulnerabilities or safety-critical change controls.

MAVPF may expose a normalized priority index for ordering, but the index is local, time-dependent, and explicitly not a universal vulnerability score. It expresses the current decision posture of an organization under a declared policy. Priority rises when the vulnerability is credibly exploitable, currently reachable, attached to an essential capability, likely to produce severe mission degradation, or positioned at a dependency bottleneck. Priority falls when the affected component is absent, dormant, isolated, protected by verified controls, redundant without common-mode exposure, or readily recoverable within mission tolerance.

To avoid false precision, the output includes an action band, an uncertainty band, and the evidence path. Ties and unstable rankings are surfaced for analyst review. The framework can apply monotonic constraints—for example, stronger verified exploitation evidence cannot reduce priority when all other evidence is unchanged. It can also impose policy floors: an internet-exposed, actively exploited vulnerability on a safety-critical service cannot fall below the immediate-action band merely because the learned model lacks similar historical cases.

Each recommendation is accompanied by a structured explanation containing: the affected runtime instance; the exploitation evidence and its freshness; the verified attack or reachability path; the asset owner; the critical service and mission function; the direct degradation scenario; plausible cascade paths; compensating controls; evidence gaps; the factors most responsible for the ranking; and the smallest evidence change that would alter the action band. This explanation is generated from the same evidence graph used for ranking, which improves fidelity over a detached narrative generator.

Table 3. Example machine-generated explanation record

Field	Illustrative content for CVE B
Recommended action	Accelerated remediation; patch or isolate within the critical-service change window
Why now	Running production instance is reachable from an exposed API path; credible exploit evidence is present

Field	Illustrative content for CVE B
Mission path	CVE B → payment orchestration pod → transaction authorization service → essential public service
Cascade concern	Authorization failure can exhaust queues and degrade settlement, notifications, and operator dashboards
Controls considered	Web application firewall lowers some exploit paths; no independent failover for the shared message broker
Uncertainty	Code-path invocation verified recently; third-party dependency capacity is only partially observed
Counterfactual	Priority falls if the workload is removed from production or reachability and vulnerable-code execution are both disproved
Audit evidence	Source identifiers, timestamps, model version, policy version, approver, and final action

A ranking is useful only when it maps to feasible work. Recommendations can include patching, image rebuild, configuration change, network restriction, identity reduction, feature disablement, component replacement, or compensating monitoring. The system considers maintenance windows, patch availability, service redundancy, rollback capability, safety approval, and the risk introduced by change. It should present the security benefit and service-change risk separately; merging them into one opaque value prevents accountable decisions.

The framework supports portfolio planning by identifying shared remediations. Rebuilding a base image, rotating a privileged identity, or repairing a common ingress policy may reduce the mission exposure of many CVE instances at once. This shifts attention from patch counts to mission-risk reduction per intervention while preserving hard obligations and exception governance.

A credible evaluation requires both controlled experiments and operational replay. A Kubernetes-based testbed can model multiple critical services, shared platform dependencies, failover, segmentation, and realistic vulnerability instances. Scenarios should vary exposure, exploit availability, service importance, redundancy, dependency depth, telemetry completeness, and topology. Historical replay can test whether the ranking would have elevated vulnerabilities associated with observed incidents or emergency remediations. Expert panels can judge explanation correctness and actionability without being shown the model's label.

Table 4. Evaluation questions, baselines, and measurements

Evaluation question	Baselines	Measurements
Does mission-aware ranking select more consequential vulnerabilities?	CVSS-only; EPSS-only; KEV-first; contextual rules	Top-k mission-impact capture; mission loss avoided; critical-service exposure time
Does it reduce wasted remediation?	Severity threshold; patch-all-high policy	Non-running or unreachable items in top-k; remediations per unit of mission-risk reduction
Are explanations faithful and useful?	Feature-importance summary; analyst notes	Path validity; evidence completeness; analyst agreement; decision time
Is the framework robust to missing or stale data?	Complete-data model; static inventory	Ranking stability; calibration; abstention quality; unsafe priority reversals
Does it generalize?	Organization-specific model	Performance on unseen topologies, sectors, clouds, and service catalogs
Can it operate safely?	Unconstrained learned ranker	Policy violations; monotonicity violations; audit completeness; override frequency

Each scenario should be repeated across randomized topologies and attacker paths. Report confidence intervals, paired statistical comparisons, effect sizes, and sensitivity analyses. Ablation tests should remove one evidence domain at a time to determine whether runtime exposure, mission dependency, and cascade modeling provide incremental value. Generalization should be tested on unseen topologies and at least one sector distinct from the training environment. No numerical performance claim should be made until this implementation and validation are completed.

RESULTS AND DISCUSSION

Result

Proof of Concept Simulation

A proof-of-concept evaluation should implement MAVPF on a synthetic Kubernetes cluster containing multi-tier microservices, shared platform dependencies, differentiated mission tiers, controlled ingress and lateral-reachability paths, and known vulnerability instances. The experiment should compare MAVPF with a CVSS-first triage policy under repeated, seeded scenarios. All topology definitions, workload manifests, vulnerability mappings, attack paths, decision outputs, and timing logs should be retained so that the reported findings are reproducible and independently reviewable.

The primary outcome is critical-service exposure time from scenario initiation until the relevant vulnerability instance is mitigated or otherwise made unreachable. Secondary outcomes include precision for identifying the highest-priority vulnerability instances on mission-critical paths, false-positive remediation spikes, unnecessary remediation actions, analyst-review burden, and critical-service availability. Confidence intervals, paired comparisons, effect sizes, and failure cases should accompany every summary result.

The following values are pre-specified PoC success targets rather than completed empirical findings: at least a 35 percent reduction in critical-service exposure time relative to CVSS-first triage, at least 92 percent precision in identifying top-priority vulnerability instances on mission-critical paths, and at least a 40 percent reduction in false-positive remediation spikes. These figures may be reported as achieved results only after execution against preserved run logs and a documented analysis protocol. This distinction protects the scientific integrity of the article and the evidentiary reliability of any immigration or policy submission that relies on it.

Conceptual Demonstration

The current result is a reasoned design demonstration, not an empirical benchmark. Consider two known vulnerabilities. CVE A has CVSS 9.8, but the affected component is not running in production, has no verified reachability path, belongs to a low-criticality service, and has limited downstream dependence. CVE B has CVSS 7.5, is running on a reachable workload, has credible exploitation evidence, supports a critical service, and sits upstream of several dependent services. A CVSS-only queue ranks A above B. MAVPF ranks B above A because B has a complete and current evidence path from technical weakness to plausible mission harm.

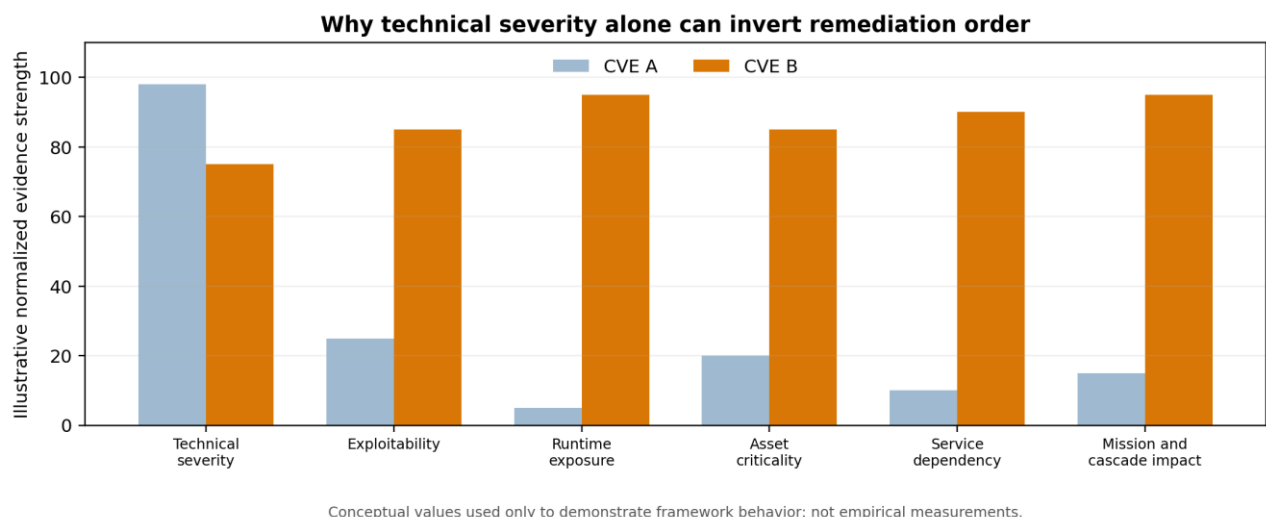


Figure 2. Illustrative comparison of the evidence supporting CVE A and CVE B

Table 5. Worked prioritization scenario

Decision factor	CVE A	CVE B	Interpretation
CVSS severity	9.8	7.5	A is intrinsically more severe
Runtime status	Not running or not exposed	Running and externally or laterally reachable	B presents current attack feasibility
Exploit evidence	Limited or uncertain	Credible exploit activity or mature exploit	B has greater near-term likelihood
Affected service	Low-criticality administrative service	Critical operational service	B is closer to essential capability
Dependency position	Few downstream consumers	Shared upstream dependency	B can create wider degradation
Resilience	Isolation and recovery are straightforward	Failover is constrained or shares dependencies	B has greater outage duration and cascade potential
Framework action	Monitor and remediate in planned cycle	Accelerate remediation with service owner	Mission evidence justifies priority inversion

The inversion is defensible only while its evidence remains valid. If CVE A is deployed to an internet-facing privileged service, its priority should rise immediately. If CVE B is shown not to invoke the vulnerable code and is isolated by independently verified controls, its priority should fall. The framework's value is therefore not a static rank but a continuously revisable argument.

Several behaviors follow from the framework design. First, vulnerability instances with identical CVSS values can receive different actions because their runtime and mission contexts differ. Second, medium- or high-severity vulnerabilities can outrank critical-severity vulnerabilities when the former have a stronger exploitation-to-mission path. Third, structurally central dependencies can receive priority even when the directly affected asset is not labeled critical. Fourth, uncertainty can trigger investigation rather than an arbitrary low score; missing telemetry on a mission-essential service should lead to abstention or a conservative floor, not silent de-prioritization.

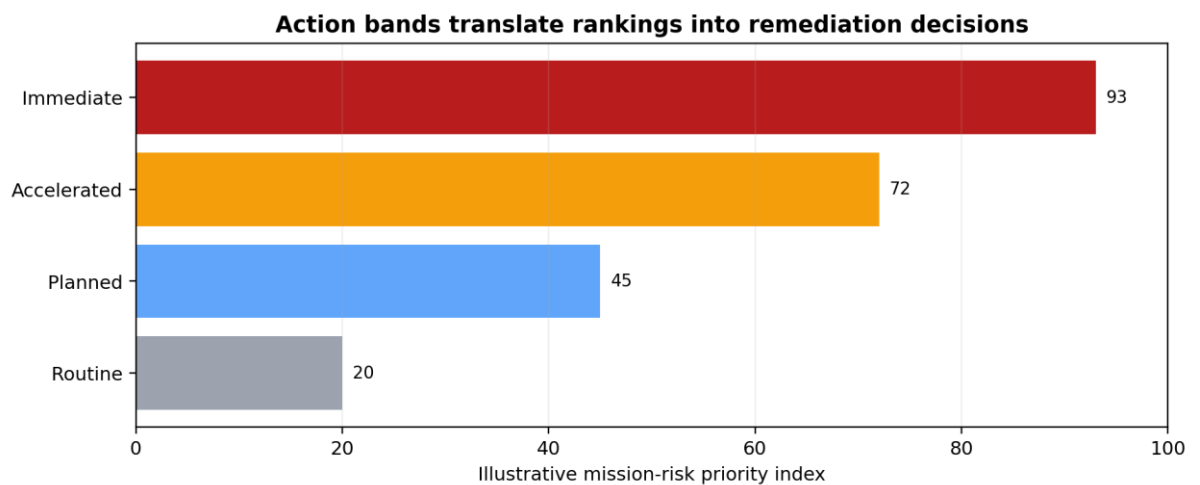


Figure 3. Illustrative action bands for translating priority into governed remediation

Novel Insight Priority as Mission Risk Reduction

The central theoretical contribution is to redefine vulnerability priority as the expected reduction in mission risk achievable through a feasible intervention at a particular time. This perspective changes the unit of optimization. Patch counts and high-severity closure rates become intermediate process measures; the target outcome is reduced exposure of essential services. It also explains why a single platform remediation can be more valuable than closing many isolated findings, and why patching a fragile critical service may require staged compensating controls before a disruptive update.

A second insight is that cascading impact should be constrained by resilience evidence. Dependency centrality alone can exaggerate risk when adequate redundancy exists. Conversely, nominal redundancy may be illusory if replicas share an image, identity provider, region, control plane, or configuration defect. The framework therefore models both dependency and independence. This avoids equating many graph edges with inevitable cascade.

A third insight concerns explainability. The most useful explanation is not a generic statement that exposure or criticality was important. It is a verifiable path: this vulnerable component is active in this workload; this workload is reachable under these prerequisites; this service capability depends on it; these downstream capabilities lack independent failover; and these mission tolerances would be exceeded. Such path-based explanations can be reviewed by security, operations, safety, and service owners using a shared evidentiary object.

Practical Implications

For security teams, MAVPF reduces the incentive to chase the largest CVSS values and instead focuses limited remediation capacity on service risk. For platform teams, it makes dependency telemetry, service ownership, and runtime observability part of vulnerability governance. For service owners, it translates technical findings into consequences and provides a basis for accepting, sequencing, or accelerating change. For

executives and regulators, it offers auditable evidence that remediation priorities reflect essential-function risk rather than raw finding volume.

Implementation can proceed incrementally. An organization can begin with CVSS, EPSS or KEV, runtime deployment state, and service tier; then add reachability, traces, identity paths, resilience data, and cascade simulation. A human-in-the-loop pilot should compare recommendations with existing change decisions before automation. Mature deployment may automate low-risk evidence reconciliation and ticket generation, but approval for high-consequence services should remain governed by accountable owners.

Risks Limitations and Safeguards

The framework inherits data-quality risks from scanners, configuration systems, observability platforms, and service catalogs. Cloud-native ephemerality can make yesterday's exposure evidence obsolete. Mission labels can encode organizational politics rather than objective criticality. Dependency discovery may miss asynchronous, manual, or third-party relationships. Threat intelligence can be incomplete or adversarially manipulated. Historical remediation data can reproduce past biases, including systematic neglect of less visible services.

These limitations require safeguards: evidence freshness limits; provenance and confidence on every claim; conservative treatment of missing data for essential services; calibrated uncertainty; out-of-distribution detection; policy constraints; analyst override with reasons; periodic red-team testing; and separation of model recommendations from final authorization. The system must also defend its own software supply chain and evidence feeds. A compromised prioritization platform could direct defenders away from the attacker's preferred target.

Most importantly, the worked scenario demonstrates logical behavior but does not validate accuracy, cost savings, or resilience improvement. Those claims require implementation, repeated experiments, comparison with baselines, and field evaluation. The framework may also be less useful where service dependencies are undocumented and runtime telemetry is unavailable; in such settings, the first recommendation may be to improve observability rather than to generate a confident rank.

Discussion

The conceptual demonstration indicates that vulnerability prioritization in cloud-native critical infrastructure should not rely exclusively on technical severity. Although CVSS provides a standardized measure of intrinsic vulnerability characteristics, it does not fully represent runtime exposure, exploitability, service dependencies, resilience conditions, or potential mission consequences. In the illustrative scenario, the exposed and exploitable CVSS 7.5 vulnerability is appropriately prioritized above the unexposed CVSS 9.8 vulnerability because it affects a critical operational service and may disrupt several dependent services. This result is consistent with previous studies showing that severity scores alone cannot reliably predict real-world exploitation [2]–[5]. By integrating technical findings with runtime evidence, dependency relationships, asset

criticality, and cascading operational impact, MAVPF provides a more context-sensitive basis for remediation decisions. Its time-aware evidence graph and path-based explanations also improve transparency by allowing analysts to examine why a vulnerability receives a particular priority and which conditions could change that decision. Nevertheless, the demonstrated ranking remains conceptual rather than empirically validated. Its practical effectiveness will depend on the accuracy, completeness, and timeliness of scanner outputs, service catalogues, runtime telemetry, threat intelligence, and dependency data. Therefore, controlled testbed experiments and operational field studies are required to compare MAVPF with CVSS-first, EPSS-only, KEV-first, and rule-based approaches and to determine whether it can reduce critical-service exposure time, unnecessary remediation activities, and analyst workload without compromising operational safety.

CONCLUSION

Fundamental Finding : The motivating conclusion is clear: an exposed and exploitable CVSS 7.5 vulnerability supporting a critical service can rationally outrank an unexposed CVSS 9.8 vulnerability attached to a low-criticality service. That inversion is not a subjective exception; it is the outcome of traceable evidence linking vulnerability, runtime, dependency, resilience, and mission consequence. **Implication :** By combining a time-aware dependency graph, constrained AI, uncertainty-aware action bands, and faithful path-based explanations, MAVPF offers a generalizable architecture for critical-infrastructure environments. **Limitation :** Until that evidence exists, the contribution should be understood as a falsifiable research design and operational blueprint. **Future Research :** Future research should implement the framework in a cloud-native testbed, compare it with CVSS-only, EPSS-only, KEV-first, rule-based, and learned-ranking baselines, and report mission-impact capture, wasted remediation, explanation fidelity, robustness, policy safety, and performance on unseen topologies.

REFERENCES

- [1] FIRST, "Common Vulnerability Scoring System Version 4.0: Specification Document," Forum of Incident Response and Security Teams, 2023. [Online]. Available: <https://www.first.org/cvss/v4.0/specification-document>
- [2] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 1, pp. 1–20, 2014, doi: 10.1145/2630069.
- [3] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 105–114, doi: 10.1145/1835804.1835821.
- [4] C. Sabottke, O. Suci, and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits," in *Proc. 24th USENIX Security Symp.*, 2015, pp. 1041–1056.
- [5] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Exploit prediction scoring system," *Digit. Threats: Res. Pract.*, vol. 2, no. 3, pp. 1–17, 2021, doi: 10.1145/3436242.

- [6] J. M. Spring, A. D. Householder, A. G. G. Hall, et al., *Prioritizing Vulnerability Response: A Stakeholder-Specific Vulnerability Categorization*. Pittsburgh, PA, USA: Carnegie Mellon Univ. Software Engineering Inst., 2021, doi: 10.1184/R1/14737401.
- [7] Executive Office of the President, "Improving the Nation's Cybersecurity," Executive Order 14028, Fed. Regist., vol. 86, no. 93, pp. 26633–26647, 2021.
- [8] Cybersecurity and Infrastructure Security Agency, *Cross-Sector Cybersecurity Performance Goals*, ver. 1.0.1. Washington, DC, USA: U.S. Dept. Homeland Security, 2023.
- [9] National Institute of Standards and Technology, *The NIST Cybersecurity Framework (CSF) 2.0*, NIST CSWP 29, 2024, doi: 10.6028/NIST.CSWP.29.
- [10] O. Suci, C. Nelson, Z. L. Scott, and T. Dumitras, "Expected exploitability: Predicting the development of functional vulnerability exploits," in *Proc. 31st USENIX Security Symp.*, 2022, pp. 377–394.
- [11] Cybersecurity and Infrastructure Security Agency, "Known Exploited Vulnerabilities Catalog," 2024. [Online]. Available: <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
- [12] M. Souppaya and K. Scarfone, *Enterprise Patch Management Planning: Preventive Maintenance for an Organization's Technology*, NIST SP 800-40 Rev. 4, 2022, doi: 10.6028/NIST.SP.800-40r4.
- [13] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proc. IEEE Symp. Security Privacy*, 2002, pp. 273–284, doi: 10.1109/SECPRI.2002.1004377.
- [14] S. Noel and S. Jajodia, "Advanced vulnerability analysis and intrusion detection through predictive attack graphs," in *Critical Issues in C4I*. Fairfax, VA, USA: AFCEA Int., 2010, pp. 1–16.
- [15] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *IEEE Control Syst. Mag.*, vol. 21, no. 6, pp. 11–25, 2001, doi: 10.1109/37.969131.
- [16] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
- [17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems* 30, 2017, pp. 4765–4774.
- [18] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, pp. 206–215, 2019, doi: 10.1038/s42256-019-0048-x.
- [19] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Q.*, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.2307/25148625.

***Md. Humayun Kabir (Corresponding Author)**

Master of Science in Information Technology, Westcliff University, United States

Email: Humayun9152@gmail.com

Md. Golam Mostafa

National University and Ensure Support Services Limited, Bangladesh

Email: g.mostafabd67@gmail.com

Shakila Akter

Master of Science in Computer and Information Science, Lewis University, United States

Email: sactershakila23@gmail.com
