Iqra Journal of Engineering & Computing
Vol. 02, No. 1, (March 2026) 12 – 35
DOI 10.21621/ijec.20260101.02

From Vulnerability Severity to Security Debt: An Evidence-Driven Prioritization Model for OWASP 2025 Web Application Risks in Low-Resource Healthcare and SME Systems

Muhammad Shahzad Khadim[1*]

ABSTRACT

Web application security reports are usually ordered by technical severity, but low-resourced organizations want to know which vulnerability will become the most expensive security debt to repay. This paper presents SECURE-DEBT, a model used to rank the 2025 OWASP web application risks to prioritize the remediation of web applications in resource-constrained healthcare-style organizations and small to medium enterprises (SMEs). SECURE-DEBT considers the unresolved vulnerabilities as security debt, thus, incurring future risk based on evidence, exposure, sensitive information, exploitability, operational impact, and time considerations for a fix. The study follows an empirically-driven, framework-based conceptual research article. The scoring considers mapping to OWASP Top 10:2025, collecting penetration-testing evidence using manual and tool-based approaches, defining factor scores, and calculating the Security Debt Priority Score. The paper then combines severity, exploitability, evidence confidence, sensitivity, operational impact, exposure, and risk of delay, while subtracting ease of remediation, to produce the security debt score. The paper compares the security debt metric to more customary High/Medium/Low reporting and CVSS-type severity scores. The paper contributes a practical, transparent and auditable model to help organizations who do not have the resources to remediate all vulnerabilities immediately, prioritize vulnerabilities for remediation in a defensible manner. The framework could be used by penetration testers, developers, healthcare application owners, and SME decision makers to relate technical findings to remediation decisions, business continuity and technical debt.

Keywords: Security Debt; Web Application Security; OWASP 2025; Vulnerability Prioritization; Penetration Testing

Author's Affiliation:
Institution(s) Name: [1]Kohat University of Science and Technology, Kohat, Pakistan
Country:
Corresponding Author's Email: [11]shahzadkhaaddim257@gmail.com

Published by ORIC, Iqra University, Main Campus, Karachi-Pakistan.

3105-4528 (Online), 3105-451X (Print) © 2026.

1. Introduction

Web apps are now used for daily services like appointment booking, emergency care coordination, school portals, small business ops, e-commerce and document processing and customer support. Since they are cheap to adopt and entry points from standard browsers most healthcare-type systems and SMEs rely on web platforms. But this convenience also introduces an exposed attack surface. All authentication pages, role-specific dashboards, API endpoints, upload forms, password reset workflows, database queries or even errors handling libraries could become entry points for intruders. The OWASP Top 10 is probably the most recognized starting point awareness document on the top 10 application security risks facing today's web applications [1] This 2025 version updates that taxonomy to match current patterns in web application risk.

An SME, local emergency service, education portal or small-scale clinic does not have charging power; one developer sharing the hosting of a single site with sparse documentation and testing performed every so often. The main security problem in such environments is that it should not observe vulnerabilities. The bigger issue is that known vulnerabilities go unmitigated because there is no defensible means by which the organization can prioritize remediation.

Standard penetration-testing reports typically rank findings as critical, high, medium or low. Severity labels help quickly communicate technical seriousness. However, severity by itself does not answer whether it should be fixed over another issue under precisely the same business rules of a particular organization.

This paper introduces the concept of security debt a term used to refer to residual weaknesses in computer security that build up over time, imposing risk, cost, and operational overhead. It is related to the more general technical debt metaphor, which presents short term engineering compromises as maintenance costs in the future [14],[15]. For security especially this is deadly for debt because the interest is not just slower development; it may also mean exposing your organisation to breaches, regulatory consequences and emergency patching but can cover reputational damage as well as loss of service availability. Security debt compounds for low-resource organizations when remediation is deferred due to teams not knowing which vulnerabilities are most likely to impact their systems.

The paper produces four meaningful outputs. For one, it establishes a structured process for gathering and verifying evidence of vulnerabilities. Second, it maps findings to OWASP 2025 categorizations so that reports still use familiar web security terminology. Third, it proposes a clear and transparent score (a cut-off for determine the range of the scores) along with priority bands. Fourth, it provides a case-study evaluation methodology that can be used on authenticated healthcare-type, SME-style and open-source benchmark laboratory web apps like OWASP Juice Shop or DVWA.

The present manuscript is prepared as a framework-based conceptual research article with an illustrative evaluation protocol. It provides a complete prioritization model, methodology, evaluation logic, tables, and figure-ready diagrams. The manuscript does not claim completed experiments or actual measured results. Illustrative examples are

included only to demonstrate how the model can be applied in future authorized case-study validation.

2. Background and Conceptual Foundations

Web application security involves protecting application logic, user identity, session management, data processing, server configuration, APIs, client-side behavior, and dependency chains. Common weaknesses include weak authentication, insecure access control, SQL injection, cross-site scripting, unsafe file upload, insecure deserialization, information leakage through errors, and missing logging. The OWASP Web Security Testing Guide provides a structured knowledge base for testing applications and web services, while the OWASP Application Security Verification Standard offers requirements that can be used to evaluate security controls [2], [3].

For low-resource systems, web application security is often implemented informally. A developer may focus on functionality and deploy the application quickly because business or service pressure is high. Security controls may be added after release, and vulnerability reports may arrive when the team already has a backlog of feature requests. In this environment, a long report is not enough. The organization needs a ranked, evidence-supported repayment plan for security debt.

The OWASP Top 10:2025 list is composed of the following articles: Broken Access Control, Security Misconfiguration, Software Supply Chain Failures, Cryptographic Failures, Injection (Common & Avoidable), Insecure Design, Authentication Failures(Common & Avoidable), Software or Data Integrity Failures(Important but avoidable), Security Logging and Alerting Failures(Important but avoidable) & Mishandling Exceptional Conditions [1]. The 2025 taxonomy is helpful to SECURE-DEBT since it arranges findings that are intelligible for application owners and developers categories of underlying symptoms accompany the methods and arguments utilized. It also highlights key changes in risk when it comes to web applications: exposure across the supply chain, lack of logging and alerting, and handling of exceptional conditions.

But OWASP categories are not sufficient for a full-blown prioritization model. A category does not say that an organization ought to remediate a riskiest issue before another. An A01 Broken Access Control finding identifying an open access to patient records or administrative functions is critical while another finding in a non-sensitive staging feature could be lower priority. SECURE-DEBT takes an extra layer for classification relying on OWASP 2025 and finally incorporates evidence, context and delay analysis.

Security debt is effectively a record of build-up of unresolved security deficiencies, in which its total price raises as associations postpone fixing. Security debt has adversarial uncertainty, which distinguishes it from standard backlog items. A weakness that an attacker might find before the organization pays off the debt. Dependency may be easier to exploit once the public proof of concept code drops. May not be harmful at present, but if some other exploitable issue exists, the absence of a log may stop us from being able to detect an incident. They are examples of how security debt can be considered from a risk-management perspective rather than just another technical backlog item.

The debt notion also aids in communicating with non-technical stakeholders. You won't go into IDOR vs broken object-level authorization, since a chief executive, clinic manager or SME owner has no clue what you are talking about and probably will never understand; but they can comprehend that leaving weaknesses unattended is a liability foreshadowing future battle. Hence, the model converts the technical information into high-priority bands like Critical Security Debt, High Security Debt, Moderate Securities Debt, Low Securities debt, and Monitor Only.

Eliminate security incidents affecting the sales process in SMEs, destroy customer trust and force them to bear recovery costs that can be more than remediation costs not previously incurred.

Careers in these environments require pragmatic models that are a little more reusable without a robust security operations center. SECURE-DEBT is tailored for this context. It does not offer a data feed, it also can be to high level machine learning. That involves rigorous evidence collation, human validation, contextual scoring and a clear rationale for each remediation priority.

Table 1: Comparison of Existing Prioritization Approaches and SECURE-DEBT

Approach	**Primary Focus**	**Strength**	**Limitation Addressed by SECURE-DEBT**
Traditional penetration-testing severity	High/Medium/Low technical finding labels	Simple and easy to communicate	Often lacks evidence confidence, delay risk, and business context
CVSS-style scoring	Standardized vulnerability severity characteristics	Useful for consistent technical communication	CVSS base severity should not be used alone as organizational risk
EPSS-style exploit prediction	Probability of exploitation in the wild for CVEs	Adds exploitation likelihood where CVE data exists	Not every web app finding has a CVE or public exploit data
NIST CSF-oriented risk management	Governance, prioritization, communication, and risk processes	Strong enterprise risk language	Needs application-level operationalization for web pentesting findings
SECURE-DEBT	Evidence-based security debt priority for OWASP 2025 web risks	Combines evidence, context, exposure, delay risk, and remediation simplicity	Requires careful human judgment and validation

Table 2: Key Concepts and Variables Used in SECURE-DEBT

Concept	Definition in This Study	Scoring Relevance
Technical severity	Intrinsic technical seriousness of the weakness	Risk-raising factor
Exploitability	Ease with which the weakness can be abused in the tested context	Risk-raising factor
Evidence confidence	Strength and repeatability of request-response, screenshot, log, or manual proof	Risk-raising factor when confidence is high
Data sensitivity	Value and sensitivity of data affected by the weakness	Risk-raising factor
Operational impact	Potential effect on service continuity, workflow, or safety	Risk-raising factor
Exposure level	Degree to which the affected component is internet-facing, user-facing, or widely reachable	Risk-raising factor
Delay risk	Expected growth in risk if remediation is deferred	Security debt multiplier concept
Remediation simplicity	Ease, cost, and speed of fixing the weakness	Subtracted because simple fixes should reduce debt quickly

3. Related Work and Literature Review

Most web application security assessments are based on the Open Web Application Security Project (OWASP) because they provide accessible categories, testing guidance, and some standard for defining or verifying fixes. Injection, cross-site scripting, broken authentication, access control failures, misconfiguration and insecure components are the common ones covered in many studies and professional assessments. The value of this body of work is that it provides a common language for testers and helps developers identify weakness patterns. While many OWASP-based assessments are limited to detection or category-level discussion. They do not always include a means for prioritizing which problem should be repaired first within the limits of budget, timeline and personnel.

SECURE-DEBT is built on OWASP rather than replacing it. OWASP Top 10:2025 is used as a baseline for classification and OWASP WSTG [1], [2] is established as an accompaniment to testing on the model. Its value add comes in the form of a debt focused prioritization layer post evidence collection. The framework moves from awareness and detection towards decision support in this way.

CVSS v4.0 Based on the common framework, is a well-known and widely used scheme to communicate properties of software vulnerabilities and severity. It contains metric groups Base, Threat, Environmental and Supplemental metrics [8]. NVD also introduces

CVSS as a means to provide a qualitative measure of severity, differentiating between severity and risk [9]. EPSS works alongside severity scoring, providing an estimate of the probability that a CVE will be exploited in production systems in the near-term [10]. Additionally, CISA maintains the Known Exploited Vulnerabilities catalog to help defenders identify vulnerabilities with known exploitation activity [11].

These tools are very powerful, but the finding of web application penetration tests may sometimes include custom application logic flaws that do not have a CVE identifier. A local healthcare portal IDOR for example, would not be in the NVD nor contain an EPSS value nor appear in CISA KEV But it could also reveal sensitive information and demand urgent remediation. To bridge this gap, SECURE-DEBT uses evidence confidence, data sensitivity, operational impact, and delay risk as locally assessable features.

Some findings may go unaddressed because the owners of the affected systems are unable to tell which issues require immediate attention and which are simply routine hardening tasks. Thus, they form a queue of pending vulnerabilities bit by bit, turning into security debt. As a result, to both technical and managerial stakeholders, a decision-support model must be transparent.

Technical debt research show that short-term software decisions may lead to maintenance burden in the future [14], [15]. A specific case of this issue is security debt, in which unresolved vulnerabilities, ineffective controls, outdated components and absent monitoring create a backlog of accumulated risk. The term has been increasingly used to describe ongoing unaddressed defects and limited capacity for remediation in industry reports [12], [13]. While this definition may vary from industry to industry, the general idea is that security work postponed is risk and expense realized later.

This is a web-application penetration-testing-level implementation of the concept of security debt called SECURE-DEBT. It views debt not as a fuzzy metaphor, but as a scoreable decision construct with evidence and exposure tied to delay in or remediation of the debt.

Most of the literature and practice around vulnerability detection, mapping with OWASP, communication of CVSS severity level, prediction for exploitability for CVEs; and Cybersecurity risk management is already well guided. The gap is the lack of a public, data-driven security debt prioritisation model for low resource healthcare and SME systems that targets OWASP 2025 web application risks. Current strategies are either dedicated to weakness type classification, requiring technical severity scoring or guiding risk management at the broader governance level. SECURE-DEBT bridges this gap by linking application-level evidence to debt aware remediation priority.

Table 3: Research Gap and Contribution Mapping

Observed Gap	Why It Matters	SECURE-DEBT Contribution
OWASP categories do not rank fixes by local context	Low-resource teams need to know what to fix first	Adds context and debt-based priority scoring
CVSS base severity is not full risk	Severity alone may mislead remediation planning	Combines severity with data sensitivity, exposure, and delay risk
Many web app findings lack CVE identifiers	EPSS and KEV may not apply to custom logic flaws	Uses evidence confidence and local exploitability
Penetration-testing reports may be too broad for SMEs	Teams may delay urgent fixes and accumulate debt	Produces clear priority bands and remediation roadmap
Security debt is often discussed generally	Practical application-level scoring is limited	Operationalizes security debt for OWASP 2025 findings

4. Research Gap and Problem Formulation

Penetration-testing reports provide likely vulnerability rankings by technical severity for existing web applications, yet do not reflect an adequate abstraction requisite to gauge the impact of delayed remediation to security debt in resource-constrained systems. This creates a decision gap. These new findings, they say, could result in organizations patching high bandwith but low severity vulnerabilities whilst putting off those which hold much more operational risk. Similarly, medium severity findings (which only become costly to remediate from a particular release onwards, or after the supplier development team has changed or dependency updates) can also go unwatched.

This is especially an issue in low-resource healthcare and SME systems, where staff budget and tooling are severely limited. A report might highlight 10 or more such weaknesses, but there may only be budget to fix three by the application owner immediately. If there is no transparent prioritization model, remediation may rely on gut feeling, developer preference or the common use of severity labels. We propose SECURE-DEBT as a formal solution to this problem.

The main research question is: How can OWASP 2025 web application vulnerabilities be prioritized using evidence strength, exploitability, operational impact, remediation delay, and future security debt risk?

RQ1: What is the mapping that leads to having a consistent setting for penetration-testing findings in OWASP Top 10:2025 categories?

RQ2: What are the most helpful types of evidence for strengthening a confidence level of finding within web application vulnerability?

RQ3: How to model delay risk as a pragmatic security debt factor of low resource systems?

RQ4: What are the differences between SECURE-DEBT and traditional severity-only ranking and CVSS-style ranking?

RQ5: How to use the model for remediation roadmap support for healthcare-style and SME-style applications?

The model assumes that testing is authorized, evidence is collected ethically, and scoring is performed by a competent tester or reviewer. It does not encourage testing of third-party websites without written permission. The model is not a replacement for legal compliance, clinical safety analysis, formal risk acceptance, or secure software development governance. It is a decision-support model for prioritizing web application remediation work after vulnerability evidence has been collected.

5. Proposed SECURE-DEBT Model

5.1 Model Overview

SECURE-DEBT stands for Security, Evidence, Context, Urgency, Remediation, and Exposure for Web Application Security Debt Prioritization. The model begins with application context identification, proceeds to evidence collection, maps each finding to OWASP 2025, calculates a security debt score, and produces a remediation priority band. The model is intentionally transparent. Each score is explainable and can be reviewed by another tester, developer, or decision maker.

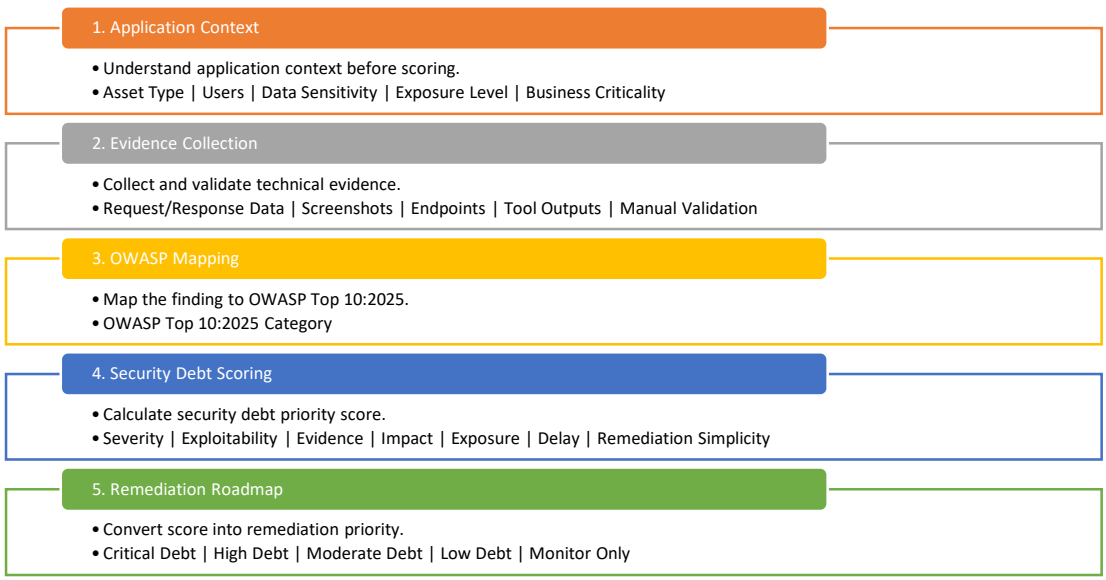

Figure 1: Conceptual Architecture of SECURE-DEBT.

The model converts web application evidence into OWASP-mapped findings, security debt scores, and remediation priorities.

5.2 Layer 1: Application Context Identification

The first layer identifies application type, user groups, sensitive data, public exposure, business or healthcare criticality, authentication model, dependency stack, and database sensitivity. This layer is essential because the same technical weakness may have different consequences in different systems. For example, weak access control on a public emergency response dashboard has a different priority than weak access control on a local test page with no sensitive data.

5.3 Layer 2: Vulnerability Evidence Collection

SECURE-DEBT is built on evidence collection. Screenshots, HTTP request-response pairs, endpoints affected, vulnerable parameter and tool output with the log observation and manual verification notes should be included to substantiate the findings. Tools can be like Burp Suite, OWASP ZAP, Nmap, SQLmap and browser developer tools or few could be manual work such a authentication test / session test log review. The model prefers documented evidence to allegations without substantiation. A weakly documented finding should not garner the same degrees of confidence as one that has been reproduced or recorded.

5.4 Layer 3: OWASP 2025 Mapping

All findings are mapped to an OWASP Top 10:2025 category. IDOR and unauthenticated object access belong to Broken Access Control. SQL injection maps to Injection. Weak password reset or credential-stuffing exposure ⇒ Authentication Failures The use of outdated packages translates to Software Supply Chain Failures. Stack trace exposure is a part of Mishandling Exceptional Conditions. Missing audit logs is mapped to Security Logging and Alerting Failures. This fixes makes sure that all reports speak the same language.

5.5 Layer 4: Security Debt Scoring

The scoring layer gives a score between 1 and 5 for each of these variables. Technical severity captures intrinsic impact. Exploitability captures attack practicality. Evidence confidence captures proof strength. Data sensitivity refers to the value of the information exposed. Operational impact describes any disruption to workflow or service. Exposure level captures reachability. Delinquency risk is the provisional debt which one expects to increment amid de-provisioning Taking away simple solutions, as these require in-pay pretty rapid and should ease burden.

5.6 Layer 5: Remediation Priority Mapping

The last layer converts the score into bands of priority. You are landing Contagious: fix it (temporary compensating controls may be desirable, if necessary). High Security Debt needs to be queued in the post remediation window. Moderate Security Debt Items already in the monitored backlog Low Security Debt may be dealt with in maintenance. Monitor

Only means we want them to be tracked but does not require a code change today unless that changes in the future context.

Table 4: SECURE-DEBT Framework Components

Layer	Purpose	Typical Evidence or Output
Application context	Understand business, data, exposure, and operational dependency	Asset profile, data classification, user roles, exposure notes
Evidence collection	Document and validate vulnerabilities	Screenshots, request-response pairs, affected endpoints, tool output
OWASP 2025 mapping	Classify findings using standard web risk taxonomy	A01-A10 mapping and CWE notes where available
Security debt scoring	Calculate debt-aware priority	Factor scores and Security Debt Priority Score
Remediation mapping	Translate score into action	Priority band, owner, recommended timeline, validation method

6. Security Debt Priority Score and Scoring Model

6.1 Formula Definition

Security Debt Priority Score In this section, we define the security debt priority score to be as follows. Security Debt Priority Score = Σ (Technical Severity + Exploitability + Evidence Confidence + Data Sensitivity + Operational Impact + Exposure Level + Delay Risk) - Remediation Simplicity They are each scored from 1-5. The higher the score, the higher the debt order. Simple problems that could be cheaply and easily fixed decrease the incentive to leave an issue unresolved so that remediation simplicity is subtracted from the total.

It is important to understand that this proposed equation is just a baseline prioritization model rather than a statistically optimized risk equation. The aim is to improve transparency, repeatability and point scoring system that make sense to lower-resourced organizations which may lack the capacity for as a higher-level forms of risk-modelling. It consequently starts with a basic additive fashioned model allowing each scoring factor to be traced, explained and examined. This is a minimal apex which can be further specified with tuned weights, interaction terms or time-variable betas in future studies as more case-study data, expert rankings or remediation outcome records are produced.

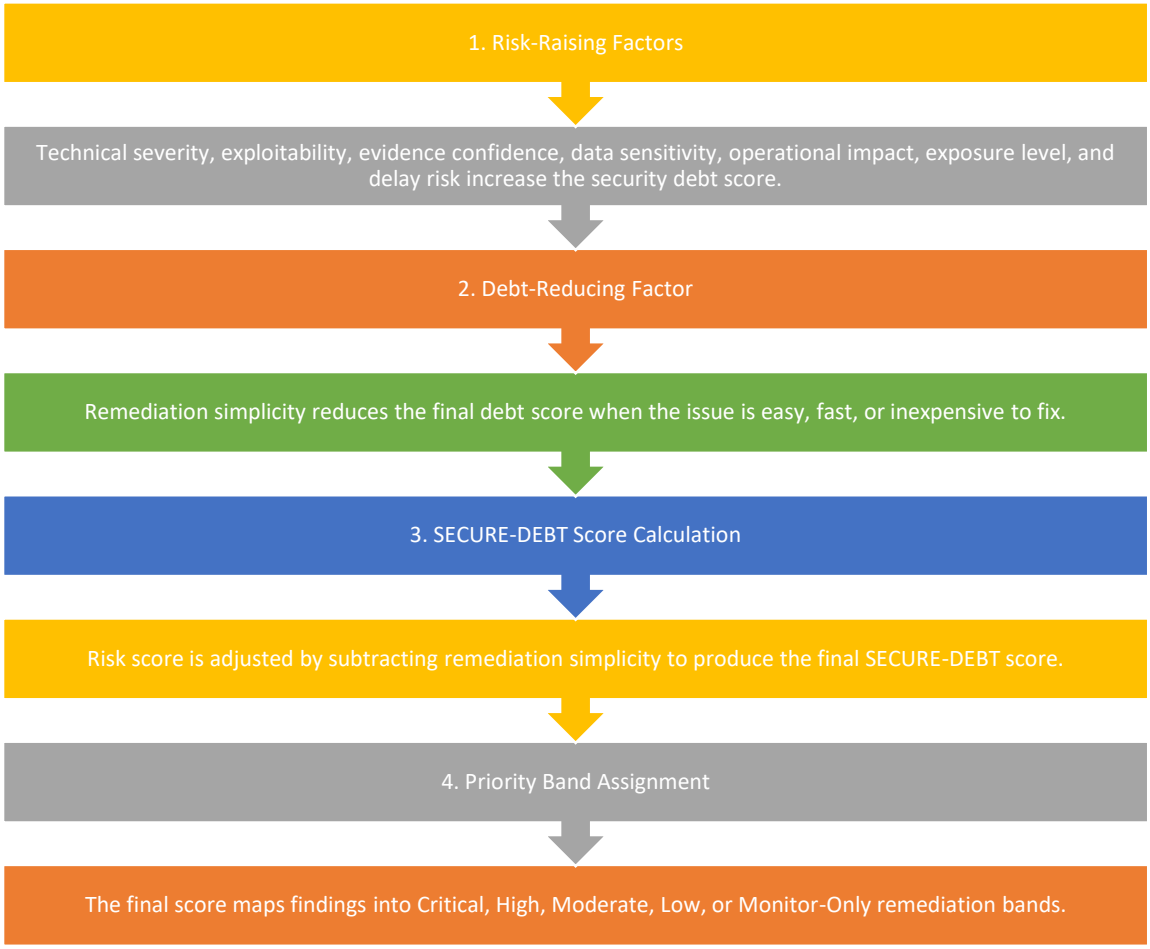

Figure 2: Security Debt Priority Score Calculation Pipeline.

The score combines risk-raising factors and subtracts remediation simplicity as a debt-reducing factor.

6.2 Scoring Scale

It uses a five-point scale: 1 = Low, 2 = Moderate, 3 = High, 4 = Critical and 5 = Severe. Not mathematically precise, but it is a consistent and audited judgement. Reviewers should keep track of why a score is given. So for example, Evidence confidence = 5 if this test issue is reproduced multiple times and there are clear examples ▾ request-response proof or it can be 2 when the tester inferred something from a scanner message without manually confirming it.

6.3 Priority Bands

The priority bands are proposed as follows: 26-35 = Critical Security Debt; 20-25 = High Security Debt; 14-19 = Moderate Security Debt; 8-13 = Low Security Debt; below eight is only monitor. Your thresholds are only to be used as a basic decision process and should

match the organizational realities. An empirical study that takes place in the future may include thresholds adjusted for expert review, remediation outcome or data from incidents.

6.4 Rationale for Subtracting Remediation Simplicity

Remediation ease is included because low-resource organizations tend to sidelining small fixes indefinitely and allowing them to become part of a larger backlog. For problems that are moderate risk, and can be easily fixed, there's little value in incurring that debt. This approach to simplicity encourages the quick payment of basic weaknesses while also allowing complex high-impact journeys to be flanked. Score this factor with caution – 5 = easy fix and a number of 1 =hard fix, expensive or architectural change required to fix.

6.5 Scoring Governance

A model based on expert judgement is in need of governance. For each score you will want to provide a brief rationale (ie reviewer name/role, date and evidence reference) In cases of disagreement, the score must be confirmed by at least one technical reviewer and one application owner. This avoids the risk of over-scoring all issues and less weighting on those which are sensitive for operations.

Table 5: Scoring Factors and Decision Guidance

Factor	Score 1 Indicates	Score 5 Indicates
Technical Severity	Limited technical impact	Severe compromise of confidentiality, integrity, or availability
Exploitability	Difficult to exploit, many preconditions	Easy to exploit remotely or with low privilege
Evidence Confidence	Unconfirmed or weak scanner signal	Manually reproduced with strong proof
Data Sensitivity	No sensitive data affected	Health, identity, credential, financial, or critical business data affected
Operational Impact	Minor inconvenience	Major service disruption, safety impact, or business interruption
Exposure Level	Internal or rarely reachable	Internet-facing or widely accessible
Delay Risk	Risk unlikely to grow quickly	Risk likely to grow through reuse, exploit publication, dependency age, or process delay
Remediation Simplicity	Difficult architectural or process change	Simple, low-cost, quick remediation

6.5 Rationale for Equal Weighting, Linearity, Variable Interaction, and Delay Risk Treatment

Equal weighting serves as an initial transparent scoring approach in the SECURE-DEBT model this should not be taken to claim that each factor is equally important in all real world environments. Note that the present study is conceptual, and no validated empirical dataset exists from which defensible statistical weights can be derived – therefore equal weightings are used. One of the main requirements for any low-resource healthcare-style and SME systems is not sophisticated mathematics, but simply a way to score things that security testers/developers and decision makers can comprehend and audit. Thus, equal weighting gives a sensible starting point which can then be refined by expert review, controlled studies or remediation history.

It follows the additive linear structure for interpretability as well. The intent of a mathematically precise breach probability by a Security Debt Priority Score is not actually the case. Rather, it offers a formal method to compare open vulnerabilities by aggregating technical severity, exploitability, evidence confidence, data sensitivity, operational impact, exposure level, delay risk, and remediation simplicity. It allows for clear contributions of each factor; a linear structure. This prevents the score from being a black-box output and allows it to be checked, in each instance, against the evidence at hand.

It is known that security risk might include interaction effects between variables. For example, an exposure can be more serious if the affected component is internet-visible. An access-control weakness in a workflow such as the one used in the healthcare domain might be operationally important where data sensitivity has been enhanced. When the delay risk is high, a vulnerability in a reused dependency may be delayed as well, so it can span across multiple releases, which makes it possibly more dangerous. While these relationships are important conceptually, the present model treats them qualitatively rather than quantitatively as an empirical calibration is beyond the scope of this framework-based approach.

To maintain simplicity and clarity, delay risk is included as an additive factor in the baseline model. Delay risk is conceptually distinct from static severity because it points to the fact that non-resolution can lead weaknesses to gain in cost, become less manageable or more exposed over time. The considered the least severe finding, a vulnerability in time of discovery may not appear very dangerous, but discounting it over and over again raises its debt burden. Versions of SECURE-DEBT that only considers time or state may represent delay risk alternatively, as a multiplier, or interaction term. Instead, the additive treatment is applied to be an explanation of fundamentals that helps form practical prioritization rather than a final mathematical model on how security debt benefits.

7. Data, Scenarios, Materials, and Case Design

7.1 Research Design

An approach for research design — experimental case-study based evaluation. In this study, author-controlled open source web applications may be used in conjunction with

intentionally vulnerable benchmark applications and third-party SME websites will only be utilized when written consent is made available. The mixed case design of the model is consistent with a realistic and generalizable approach where ethical or legal standards could not have been potentially violated.

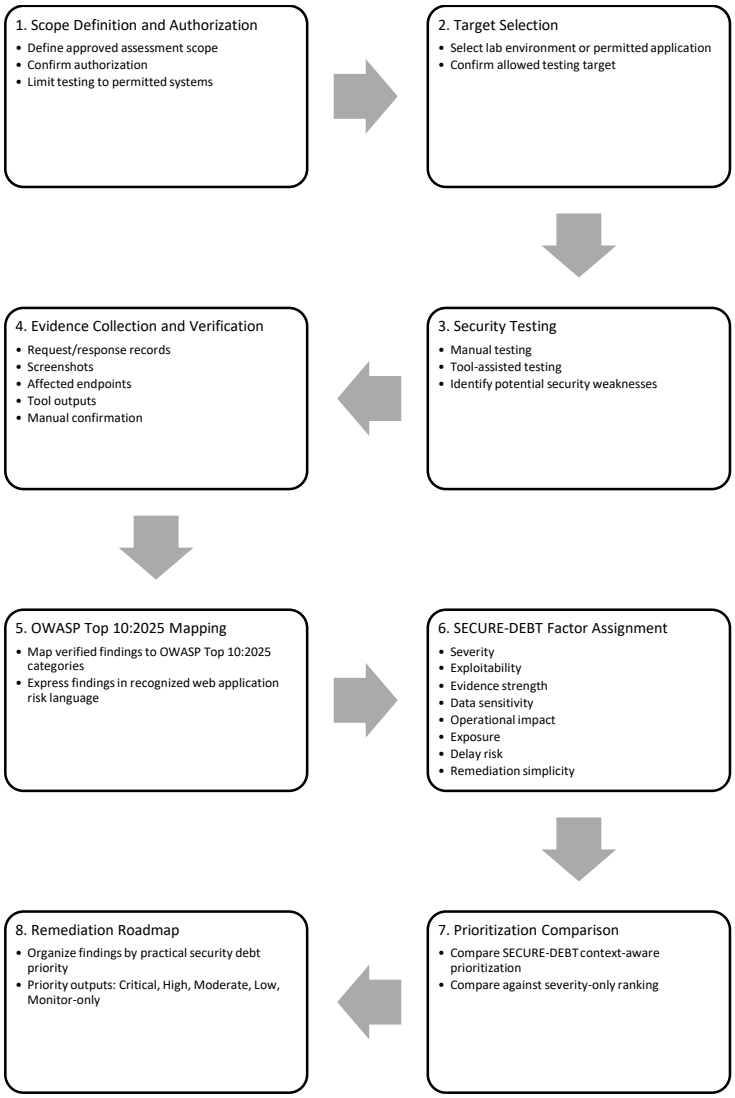

Figure 3: Research Workflow for SECURE-DEBT Evaluation.

The workflow shows authorized scope definition, testing, evidence collection, OWASP mapping, scoring, comparison, and remediation planning.

7.2 Candidate Test Applications

Candidate Applications include but are (probably;) not limited to: Emergency Care, SecureAudit, VulnTrack, SmartTask, OWASP Juice Shop, DVWA & local SME websites

with WRITTEN PERMISSION. Emergency Care and similar applications that are healthcare-style are particularly appropriate, as they typically include operational workflows and user roles and likely sensitive information. Controlled benchmark cases for demonstrating the model can be performed with a safer variant of OWASP Juice Shop and DVWA, avoiding the risk of exposing real world organizations.

7.3 Data Collection Fields

Every record should feature a unique finding number, app name, OWASP 2025 category, vulnerability type that triggers the effective endpoint, an evidence description with images if applicable and tool used along with manual validation status and final Security Debt Priority Score including factor scores (High: >80, High-Medium risk: 60–79; Medium-Low risk: 50–59), priority band (Critical, High & Medium) which determines exposure at right scan by using powerfull web UI like OWASP Application Risk Management. Also recommended Fix or any other remediation step from documents versioning is also storring as Validation method. Do not save genuine înformation sensitive record. Before publication, redact screenshots and request-response evidence.

7.4 Tools and Instruments

Kali Linux Burp Suite OWASP ZAP Nmap SQLmap Browser developer tools Manual payload testing Authentication workflow review Dependency review and Log review are some of the items on the list of Testing Toolkit. In no way should the output of a tool simply be taken as is. This is why false positives need to be manually verified as it is one way that security debt can be distorted as well as wasting limited remediation capacity.

7.5 Ethical and Legal Controls

All testing must be authorized. The study should not perform scanning, exploitation or intrusive testing of systems without express written consent. If real healthcare or SME systems are used, the consent must be clearly described in addition to scope boundaries, data minimisation, redaction and responsible disclosure. Further ethics review may be necessary if patient records, user personal data or private logs are affected. The current manuscript does not utilize or disclose any proprietary organizational data.

Table 6: Proposed Data Collection Template for SECURE-DEBT Evaluation

Field	Description	Example Value
Finding ID	Unique number for the issue	SD-001
Application	Tested system under authorized scope	Emergency Care lab deployment
OWASP 2025 Category	Mapped OWASP risk category	A01 Broken Access Control
Vulnerability Type	Specific weakness name	IDOR in record retrieval
Evidence	Proof stored in redacted form	Request-response and screenshot

Tool Used	Tool or manual method	Burp Suite and manual validation
Factor Scores	Eight SECURE-DEBT factor values	Severity 4, Exploitability 4, etc.
Security Debt Score	Final calculated score	26
Priority	Debt priority band	Critical Security Debt
Remediation Note	Recommended fix and validation method	Enforce object-level authorization and retest

8. Evaluation Method and Analytical Framework

8.1 Comparative Evaluation Design

To evaluate the effectiveness of any prioritization mechanism, we compare three approaches to ordering tickets: High/Medium/Low severity-like ranking and CVSS-style Severity-based ranking (with Low priority if a ticket has no severity) against SECURE-DEBT evidence-driven security debt ranking. This is because: it should not be supposed that one of the approaches is better than another as a universal principle. Rather, the evaluation assesses whether SECURE-DEBT generates a more executable remediation order for organizations with limited resources and therefore high levels of uncertainty by examining its consideration of evidence, data sensitivity, operational impact, exposure, and delay risk.

8.2 Evaluation Criteria

These include actionability, context sensitivity, evidence traceability, low resource usability, debt-awareness and reproducibility as well as stakeholder explainability. Actionable should question whether the output conveys what needs to be done first in an organisation. Context sensitivity is whether the (positive / negative) business and data impacts are addressed. But the evidence of that is traceability of evidence, which brings up questions of how you can link the score with evidence. Assessment of Debt queries whether delay risk and all cumulated risk has been captured. Stakeholder explainability is about if the non-expert decision maker can understand the rationale.

8.3 Expert Review Protocol

This manuscript would benefit from expert review in an empirically stronger version. For a sample of findings, this allows security practitioners, developers and application owners to individually rank their findings and compare their ranking with SECURE-DEBT outputs. Discussions about differences should refine definitions of Fr and Fj and priority bands. Although enough reviewers could be included to measure inter-rater consistency, such data are not fabricated in the current manuscript.

8.4 Illustrative Scoring Demonstration

The table below demonstrates how SECURE-DEBT scoring may work.

Table 7: Demonstrative SECURE-DEBT Scoring Scenarios

Finding	OWASP 2025 Mapping	Score Logic	Debt Band
IDOR exposes appointment details	A01 Broken Access Control	High severity, high exploitability, strong evidence, sensitive data, public user role, high delay risk	Critical Security Debt
Outdated JavaScript package with known issue	A03 Software Supply Chain Failures	Moderate immediate impact but high delay risk if dependency remains across releases	High Security Debt
Verbose stack trace on failed request	A10 Mishandling of Exceptional Conditions	Moderate severity, strong evidence, useful to attacker, simple fix	Moderate Security Debt
Missing security event alerts for admin login failures	A09 Security Logging and Alerting Failures	Low direct exploitability but high detection impact in healthcare-style workflow	High Security Debt
Clickjacking missing header on non-sensitive public page	A02 Security Misconfiguration	Low sensitivity, limited operational impact, simple fix	Low Security Debt

8.5 Validity Considerations

The threats to validity include subjective scoring, limited diversity in application, false positives from the tool used and the difficulties in estimating risk of delays. Things like inadequate amount of evidence to document, lack of reviewer calibration or segregation of examples from results can help mitigate these threats. SECURE-DEBT is approached as a decision support model instead of precise forecast for future occurrences.

9. Expected Analytical Outcomes and Evaluation Metrics

9.1 Expected Analytical Findings

In a completed evaluation, we find that for high levels of data sensitivity, exposure and delay risk, and some medium-severity vulnerabilities qualify as high security debt. Healthcare-style systems will probably continue to find themselves at the top of the hit list for flaws that are high impact based on access control (forcing their integration with user identity), sensitive records and operational workflow management. Obsolete dependencies can get a higher debt score than their immediate severity would indicate if they are reused on many pages or releases. The absence of logging may rise higher in

priority as a detection failure would present an increased time to detect and respond to incidents.

These are anticipated analytical patterns and not actual results reported in the current manuscript. They are derived from the architecture of the model and must be qualified by properly conducted case-study testing. The purpose of the model is to make such prioritisation logic explicit instead of being buried in tester's heads.

9.2 Evaluation Metrics

Key evaluation metrics include the difference in rankings, actionability rating, completeness of evidence, reviewer agreement, remediation feasibility and usefulness for backlog reduction. Comparison of SECURE-DEBT output to severity-only and CVSS-style ranking difference. Evidence completeness is about whether each finding has enough proof. Another option is that developers or application owners can rate actionability themselves. Multiple experts scoring the same findings allows for measurement of reviewer agreement. Overall clarity considers how easy or difficult it is to understand the report, including any necessary technical skills.

9.3 Remediation Roadmap Output

The delivery is a remediation roadmap. Unlike INTEGER, which delivers a longer unranked list of diseases associated with Magnusson-type states in humans — known as risk factors — SECURE-DEBT aggregates findings dependent on bands and assigns these conditioned on recommended action windows. Critical Security DebtShould Lead to Immediate Remediation or Compensating Controls High Security Debt should be assigned for the next development sprint. Moderate Security Debt is one that can be rescheduled with monitoring. Limited Risk Debt Objectives Fix on typical upkeep. Monitor Only items need to be revised whenever there are any changes in exposure, data sensitivity or threat intelligence.

9.4 Evidence-Based Reporting Benefits

As every score requires supporting evidence, evidence-based reporting can decrease false positives. It also improves communication. A developer gets full context of the vulnerable endpoint, proof of impact and a recommended fix. Manager can see why the issue builds up debt and by delaying it. An auditor reviewing security can audit the rationale for the scoring This type of structure can be especially useful in larger organisations where security expertise is more widely dispersed amongst devs, operators and business owners than consolidated under a dedicated application security team.

9.5 Interpreting Negative or Mixed Results

If SECURE-DEBT is not a significant improvement over prioritization in some instances according to empirical evaluation, that is still useful. It could suggest that factor weights need to be set differently, that some indications may not benefit from debt-based scoring or that expert reviewers would prefer a more intuitive model of the severity. These findings should be reported candidly in a ready research study. This framework is not expected to be perfect; it should rather be tested, refined and adapted.

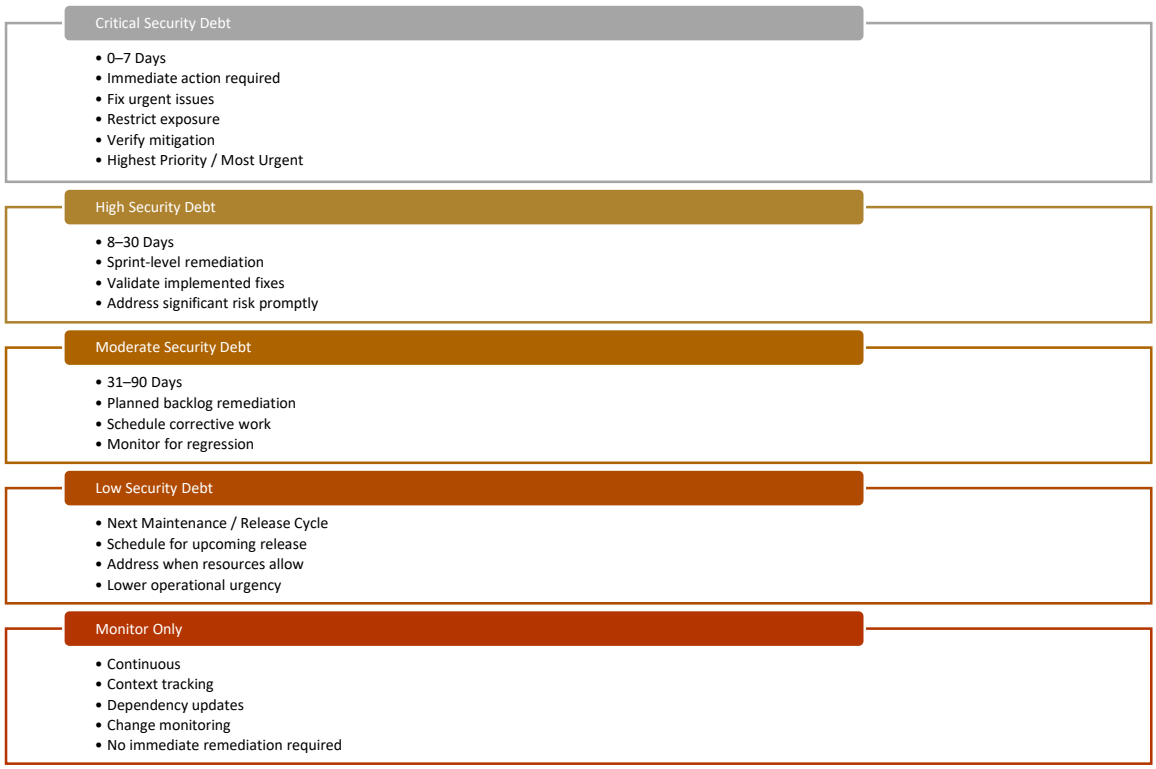

Figure 4: Remediation Roadmap from Security Debt Bands.

The roadmap translates security debt bands into practical remediation timing and management actions.

10. Discussion, Practical Implications, and Implementation Guidance

10.1 Severity Is Not the Same as Security Debt

The main message of this paper is that severity and security debt are not the same but related. Severity is a measure of the technical severity of the vulnerability. Security debt is the future burden which that weakness creates and the risk it entrains in a specific system by failing to resolve. If a serious defect is isolated, well contained and has a fix on the way soon, it may have low debt. An issue that is non-lethal but a risk to sensitive workflows and will remain exploitable without patch for months could be assigned Moderate severity with the potential for High debt.

10.2 Practical Value for Penetration Testers

SECURE-DEBT focuses on improving the way penetration testers structure reports to make the information in them more usable for clients. Testers can prioritize vulnerabilities by telling stories about how the four factors of evidence, context, exposure and

remediation delay impact prioritization instead of based solely on their severity. It helps your clients understand why a finding is important at the moment and which actions to prioritize. It also propels testers into more definitive evidence, avoiding ambiguous suggestion.

10.3 Practical Value for SMEs

SMEs rarely require a long theoretical report about security but rather step by step guidance. SECURE-DEBT can assist them in identifying the most critical fixes when budgets are restricted. It could also be used in conversations with developers, hosts or management. An unsecured debt is much easier to act on than a list of technical weaknesses. The model supports a pragmatic strategy: solve the high debt issues first, finance simple debt quickly and track low priority issues.

10.4 Practical Value for Healthcare-Style Systems

Data sensitivity, service availability and continuity of operations are critical features to account in healthcare style systems. An application that does so and serves appointments, emergency requests, patient communication or internal clinical coordination could then face significant repercussions for access control, authentication, logging or data integrity failures. SECURE-DEBT incorporates dimensions of data sensitivity and operational impact in the score, which brings these aspects into focus.

10.5 Implementation Guidance

SECURE-DEBT is a five-step process used by organizations. Define the authorized scope and context of use, first. 2nd: Test using OWASP aligned manual and tool-assisted methods. Third, preserve evidence in a structured and redacted manner. Fourth, rate each of the findings based on the specified factor scale and record your reasons for doing so. Fifth, create a remediation roadmap and validate fixes post-implementation. This model can be done in a spreadsheet, ticketing system or simple internal dashboard.

Table 8: Practical Implications for Stakeholders

Stakeholder	**Practical Use of SECURE-DEBT**	**Expected Benefit**
Penetration tester	Adds evidence confidence and debt rationale to reports	More actionable and defensible findings
Developer	Receives ranked fixes with proof and validation notes	Better sprint planning and fewer ambiguous tasks
SME owner	Sees what must be fixed first under limited budget	Improved resource allocation
Healthcare application owner	Connects security fixes to data sensitivity and service continuity	Better protection of critical workflows
Researcher	Uses the model as a testable framework	Future empirical validation and threshold refinement

11. Limitations, Challenges, and Future Research

Limitation: Currently, this manuscript provides a framework and evaluation protocol (not final empirical results). If the article is an empirical research paper, you should supply a portion of authorized case-study data with your journal submission.

SECURE-DEBT depends on expert judgment. Exploitability, operational impact or delay risk are two things great testers do differently here. This is a limiting factor that can be minimized but not negated. Improving consistency with reviewer calibration, scoring guidelines, examples and multi-reviewer discussion Future work should also assess inter-rater agreement and clarify factor definitions.

Or, a related limitation, is that the current SECURE-DEBT score employs equal weights and an additive linear formula. The design is enhanced for simplicity, transparency and auditability but this again fails to capture the relative importance of non-linear relationships among exploitability, exposure, data sensitivity, operational impact and delay risk. In the real world, one variable may be stronger than another. Such as in the case of a moderately severe weakness that is exposed to public users for an extended time frame when sensitive data might be impacted, this would quickly escalate its criticality. Subsequent validation should therefore seek to determine whether either the weighted scoring, interaction-aware models, delay-risk multipliers, or time-dependent debt functions drive better prioritization performance yields compared to the baseline model described here in.

The model is implemented in low-resource healthcare or SME ready web applications. It is possibly not directly generalizable to cloud-native enterprise systems, industrial control systems, mobile apps, or large-scale microservice platforms without adaptation. Other environments can be even more desirable than regulatory exposure, or the path complexity of a cloud identity that might need to persist through managed service disruption in addition to supply chain criticality.

Ambiguity in scope makes security testing the source of legal and ethical risks. All practical assessment should be performed on owned systems, deliberately vulnerable labs or with written consent from third-parties. When you have sensitive data, redact any evidence, responsibility on the part of researchers. Personal data should only be collected as a last resort, with the required permissions from an appropriate ethics process.

Table 9: Limitations and Future Research Directions

Limitation	Risk to Study Quality	Future Research Response
No empirical dataset in current version	Cannot claim measured effectiveness	Conduct authorized case studies and report actual findings
Subjective factor scoring	May reduce reproducibility	Use expert calibration and inter-rater analysis
Limited to web applications	May not apply to all system types	Adapt factors for APIs, mobile, cloud, and enterprise systems

Delay risk estimation is difficult	May overstate or understate debt	Use historical remediation data and threat intelligence
Low-resource focus	May not satisfy enterprise governance needs	Map outputs to enterprise GRC and risk registers

12. Conclusion

SECURE-DEBT: An evidence-driven security debt prioritization model for OWASP 2025 web application vulnerabilities, especially in resource-constrained healthcare and SME systems. This model fills both a theoretical and practical gap in how organizations approach vulnerability management, where reports are typically severity-based but there is no systematic process by which people determine which of the unpatched weaknesses will be the most destructive security debt if left unaddressed for long periods. The Security Debt Priority Score is the key output, a score that factors in technical severity, exploitability rank, evidence confidence, data sensitivity impact, operational impact and exposure extent to delay risk (and remediation simplicity). This structure assists with mapping evidence of vulnerability to bands that indicate the priority for remediation. It also makes the rationale for every priority auditable and transparent. For SMEs and healthcare-like systems, the model helps make realistic security planning in resource limited environments. It suggests that teams focus on addressing high-debt findings first, quickly pay off simpler debt, and delegate lower-priority issues. This gives penetration testers a reporting framework that we can use instead of high, medium, and low severity labels. It gives researchers a framework to test security debt focused primarily on web applications. However, SECURE-DEBT does not mathematically guarantee to predict breaches. An evidence-based, context-sensitive debt prioritization decision-support framework Future research should confirm the model empirically by using approved case studies, relate it to expert ranking, refine scoring thresholds then compare remediation results between debt-based ranking and best-practice for risk.

Funding: No external funding was declared for this manuscript.

Conflict of Interest: The author declares no known conflict of interest.

Data Availability: This framework-based study does not rely on private organizational data. The scoring examples are demonstrative and are included only to explain the proposed model. Future empirical validation will require authorized case-study datasets or controlled benchmark applications.

Ethics Statement: This study does not involve human participants, patient records, animal subjects, or private organizational data. Any future case-study testing must be performed only on owned systems or systems with written authorization.

Author Contributions: The author is responsible for conceptualization, framework design, manuscript preparation, and final review.

Acknowledgment: The author acknowledges the role of open security standards and community resources such as OWASP, NIST, FIRST, NVD, and CISA in supporting practical cybersecurity research.

References

[1] J. Li and H. Li, "Evolution of application security based on OWASP Top 10 and CWE/SANS Top 25 with predictions for the 2025 OWASP Top 10," in *Proc. 8th Int. Conf. Inventive Comput. Technol. (ICICT)*, 2025, pp. 1178–1183, doi: 10.1109/ICICT64420.2025.11004742.

[2] L. Prates and R. Pereira, "DevSecOps practices and tools," *Int. J. Inf. Secur.*, vol. 24, no. 11, pp. 1–27, 2025, doi: 10.1007/s10207-024-00914-z.

[3] M. Bedoya, S. Palacios, D. Díaz-López, E. Laverde, and P. Nespoli, "Enhancing DevSecOps practice with Large Language Models and Security Chaos Engineering," *Int. J. Inf. Secur.*, vol. 23, pp. 3765–3788, 2024, doi: 10.1007/s10207-024-00909-w.

[4] S. Nazir, S. Ahmed, H. Ullah, and A. García-Pérez, "Cybersecurity risk frameworks and compliance models for secure software systems," *IEEE Access*, vol. 12, pp. 55421–55439, 2024, doi: 10.1109/ACCESS.2024.3385417.

[5] M. Abdulsatar, H. Ahmad, D. Goel, and F. Ullah, "Towards deep learning enabled cybersecurity risk assessment for microservice architectures," *Cluster Comput.*, vol. 28, Art. no. 350, pp. 1–25, 2025, doi: 10.1007/s10586-024-05092-0.

[6] G. P. Bhandari, G. Assres, N. Gavric, A. Shalaginov, and T.-M. Grønli, "IoTvulCode: AI-enabled vulnerability detection in software products designed for IoT applications," *Int. J. Inf. Secur.*, vol. 23, pp. 2677–2690, 2024, doi: 10.1007/s10207-024-00848-6.

[7] Awan, J.H.; Pathan, S.M.; Memon, N.A.; Tahseen, S.; Ali, S.A. Proposed security model as a Predictive Tool: An approach to monitor network attacks in Pakistan. *JICT* 2025, 19, 41–46.

[8] Z. Zhang, V. Kumar, B. Pfahringer, and A. Bifet, "AI-enabled automated common vulnerability scoring from common vulnerabilities and exposures descriptions," *Int. J. Inf. Secur.*, vol. 24, no. 16, pp. 1–21, 2025, doi: 10.1007/s10207-024-00922-z.

[9] M. Malik and F. Pastore, "An empirical study of vulnerabilities in edge frameworks to support security testing improvement," *Empir. Softw. Eng.*, vol. 28, no. 99, pp. 1–41, 2023, doi: 10.1007/s10664-023-10330-x.

[10] S. Qadir, E. Waheed, A. Khanum, and S. Jehan, "Comparative evaluation of approaches and tools for effective security testing of web applications," *PeerJ Comput. Sci.*, vol. 11, e2821, 2025, doi: 10.7717/peerj-cs.2821.

[11] M. Sridharan, M. Mäntylä, and L. Rantala, "Detection, classification and prevalence of self-admitted aging debt," *Empir. Softw. Eng.*, vol. 30, no. 143, pp. 1–35, 2025, doi: 10.1007/s10664-025-10696-0.

[12] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Softw.*, vol. 29, no. 6, pp. 18–21, Nov.–Dec. 2012, doi: 10.1109/MS.2012.167.

[13] A. A. Alenezi and K. Almustafa, "Identifying the primary dimensions of DevSecOps: A multivocal literature review," *J. Syst. Softw.*, vol. 214, Art. no. 112063, 2024, doi: 10.1016/j.jss.2024.112063.

[14] M. Felderer and P. Ramler, "Web application testing—Challenges and opportunities," *J. Syst. Softw.*, vol. 219, Art. no. 112186, 2025, doi: 10.1016/j.jss.2024.112186.

[15] K. Rami, P. Halvadiya, and R. Patel, "Automatic identification of web vulnerabilities and risks in web applications," *J. Comput. Anal. Appl.*, vol. 33, no. 4, pp. 907–920, 2024.

[16] S. E. Ponta, H. Plate, A. Sabetta, M. Bezzi, and G. Daniel, "Detection, assessment and mitigation of vulnerabilities in open-source dependencies," *Empir. Softw. Eng.*, vol. 29, no. 41, pp. 1–38, 2024, doi: 10.1007/s10664-024-10412-4.

[17] Awan, J.H.; Shah, S.R.H.; Charan, K. Southeast Asia and Growing Challenge of Cyber-Attacks: A Regional Security Insight. *Asia Pac.-Annu. Res. J. Far East South East Asia* 2024, 42, 97–111.

[18] M. Felderer, P. Ramler, and R. Breu, "Advances in automated security testing for web applications," *J. Syst. Softw.*, vol. 218, Art. no. 112118, 2025, doi: 10.1016/j.jss.2024.112118.

[19] A. K. Sharma and V. Kumar, "Static and dynamic analysis techniques for detecting vulnerabilities in modern web applications," *IEEE Access*, vol. 12, pp. 44521–44539, 2024, doi: 10.1109/ACCESS.2024.3378125.

[20] N. Alshahwan, X. Gao, and M. Harman, "Automated penetration testing of web applications using intelligent fuzzing techniques," *Softw. Test. Verif. Reliab.*, vol. 35, no. 2, e1872, 2025, doi: 10.1002/stvr.1872.