UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

A Method for the Evaluation of Vulnerability Scoring Systems and the Validation of Their Underlying Statistics

João Pedro Magalhães Pires de Torgal Rolo

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Dr. Vinicius Vielmo Cogo
Prof. Dr. Alan Oliveira de Sá

2025

Acknowledgments

I would like to express my deepest gratitude to my advisors, Professor Vinicius and Professor Alan, for their invaluable guidance, insightful recommendations, and continuous support throughout this research.

I would like to extend my sincere gratitude to my company, MEO, and especially to the PRA department for their support and understanding while I carried out this research in parallel with my professional responsibilities.

Special thanks to my friends — João, Tomás, Rafael, Diogo, Filipa and Beatriz — whose encouragement and companionship were invaluable during this journey, making it lighter and more rewarding.

Finally, I would like to thank my family, especially my father and my stepmother, for their unwavering patience, love, and belief in me. Their encouragement sustained me through the most challenging moments and made this achievement possible.

In loving memory of my mother, whose love continues to guide me even in her absence.

Resumo

A gestão eficaz das vulnerabilidades constitui um dos maiores desafios de segurança da informação na atualidade. A proliferação constante de novas vulnerabilidades reportadas diariamente, associada à diversidade de contextos em que estas se manifestam e às limitações de recursos das equipas de segurança, torna impraticável a resolução imediata de todos os problemas identificados. Perante este cenário, a priorização do tratamento de vulnerabilidades revela-se uma prática essencial, permitindo que as organizações concentrem esforços na mitigação dos riscos mais críticos. Todavia, os métodos atualmente predominantes para hierarquizar vulnerabilidades — em particular os sistemas baseados no Common Vulnerability Scoring System (CVSS) — apresentam limitações estruturais que comprometem a sua eficácia. O CVSS, embora amplamente adotado, não foi originalmente concebido para suportar sozinho processos de ordenação dinâmica de vulnerabilidades segundo a probabilidade de exploração efetiva, mas antes para proporcionar uma classificação normalizada da gravidade técnica.

Neste contexto, o presente trabalho de investigação propõe e avalia um novo método de avaliação de sistemas de pontuação de vulnerabilidades, que procura aproximar a prática de priorização do risco real enfrentado pelas organizações. É introduzida uma abordagem comparativa assente na construção de uma ordenação de referência, tida como ideal, a qual é definida por dois critérios fundamentais: (i) a separação inicial das vulnerabilidades segundo os níveis qualitativos de gravidade definidos pelo CVSS (Crítico, Elevado, Médio, Baixo); e (ii) dentro de cada estrato de gravidade, a ordenação cronológica de acordo com a data de publicação mais antiga de um código de exploração público. Esta lógica assume que uma vulnerabilidade mais antiga, para a qual já existe exploração divulgada, representa um risco potencialmente mais urgente e tangível.

Com base nesta ordenação de referência, são avaliados diferentes sistemas de priorização, nomeadamente o CVSS tradicional, o Exploit Prediction Scoring System (EPSS) e, sobretudo, o SecScore, um sistema alternativo que incorpora modelos estatísticos temporais (em particular a distribuição de Laplace assimétrica) para ajustar dinamicamente a pontuação atribuída às vulnerabilidades em função da probabilidade empírica de exploração. A qualidade de cada sistema é medida através da distância euclidiana normalizada entre a ordenação produzida pelo sistema e a ordenação de referência. Para expressar a vantagem relativa de um sistema em relação a outro, é utilizado o indicador $\mathbb{Q}$, que traduz o ganho percentual na aproximação ao ideal.

A análise empírica contemplou tanto estudos de caso detalhados como uma avaliação longitudinal em múltiplas janelas temporais. Num primeiro momento, é analisado de forma aprofundada

um estudo de caso relativo à primeira semana de maio de 2005, que compreende 220 vulnerabilidades registadas. Este período foi escolhido por representar um contexto particularmente rico em diversidade de vulnerabilidades e por permitir observar o desempenho dos sistemas em condições históricas específicas. Os resultados mostraram que o SecScore apresenta melhorias significativas em relação ao CVSS nas categorias Crítico e Elevado, registando ganhos médios de aproximadamente 6 a 12% na primeira e de 17 a 20% na segunda. Estes resultados revelam a capacidade acrescida do SecScore para distinguir vulnerabilidades realmente prioritárias em contextos onde o risco de exploração é mais grave. Para a categoria Média, os resultados apresentaram uma performance mista, não evidenciando ganhos consistentes em relação ao CVSS. Já na categoria Baixa, observaram-se melhorias modestas mas mensuráveis, situando-se em torno de 1 a 2%.

Numa segunda fase, a avaliação foi alargada a um conjunto alargado de semanas consecutivas, permitindo observar a evolução do desempenho dos sistemas em termos médios e reduzir o enviesamento de análises pontuais. Os resultados confirmaram a tendência do estudo de caso, com o SecScore a apresentar ganhos médios globais de 5,2% na categoria Crítico e 6,3% na categoria Elevado, enquanto nas categorias Média e Baixa os ganhos se mantiveram mais reduzidos (cerca de 1 a 1,5%). Estes valores, embora numericamente moderados, assumem elevada relevância prática, uma vez que correspondem à diferença entre corrigir primeiro uma vulnerabilidade com elevada probabilidade de exploração versus investir recursos numa vulnerabilidade menos urgente. Assim, a vantagem do SecScore traduz-se numa alocação mais eficiente dos recursos escassos disponíveis para resposta a incidentes.

Por fim, numa última fase, é feito uma análise da influência da janela temporal utilizada para ajustar os parâmetros do SecScore. Dado que o sistema se baseia em dados históricos de exploração, o período de tempo considerado para calibrar os modelos pode afetar substancialmente os resultados. É assim demonstrado que o desempenho do SecScore tende a melhorar com o alargamento da janela temporal até certo ponto, evidenciando ganhos progressivos à medida que se incorporam mais anos de histórico. Contudo, este crescimento não é estritamente monotónico. Verificou-se que, em determinados estratos (por exemplo, a categoria Elevado), o desempenho atinge o pico em torno dos 13 anos de dados, registando melhorias assinaláveis, antes de estabilizar ou até decrescer ligeiramente. Esta constatação sugere a existência de rendimentos decrescentes na utilização de dados históricos demasiado extensos, possivelmente devido à mudança de padrões de exploração e à evolução das práticas de ataque ao longo do tempo.

Através desta análise, o trabalho reforça a ideia de que a qualidade da priorização não depende apenas da métrica em si, mas também da forma como esta é contextualizada temporalmente. O SecScore distingue-se precisamente por introduzir esta dimensão temporal e probabilística, indo além da mera avaliação estática da gravidade técnica. Este estudo aponta ainda que a integração de diferentes fontes de contexto, como a diversidade de componentes afetados e a caracterização do ecossistema em que a vulnerabilidade ocorre, constitui um caminho promissor para investigações futuras.

Assim, este trabalho propõe uma metodologia objetiva e mensurável para avaliar sistemas de

priorização de vulnerabilidades, demonstrando empiricamente, com base em dados históricos de grande escala, que é possível melhorar significativamente o processo de priorização através da adoção de métricas que incorporam consciência contextual e diversidade temporal. Para além disso, abre caminho para uma reflexão crítica sobre a utilização quase exclusiva do CVSS como padrão de facto, mostrando que, embora útil, este sistema carece de adaptações substanciais para responder às necessidades operacionais atuais.

Em síntese, este trabalho evidencia que a priorização de vulnerabilidades baseada em risco pode ser significativamente melhorada através da incorporação de consciência contextual, diversidade de componentes e modelação temporal da probabilidade de exploração. O SecScore, enquanto proposta concreta, demonstrou ganhos mensuráveis face ao CVSS e ao EPSS em múltiplos cenários, sobretudo nas categorias de maior gravidade, onde o impacto da decisão de priorização é mais crítico. Mais importante do que os resultados numéricos específicos, a investigação oferece à comunidade científica e às equipas de segurança um quadro metodológico robusto para avaliar, comparar e evoluir os sistemas de priorização. Ao sublinhar a importância de métricas adaptativas, sensíveis ao tempo e fundamentadas em dados empíricos, este trabalho contribui para aproximar a prática de gestão de vulnerabilidades da realidade dinâmica das ameaças contemporâneas.

Palavras-chave: Processo de Gestão de Vulnerabilidades, Avaliação de Risco Sensível ao Tempo, Priorização do Risco, Maturidade do Código de Exploração, Avaliação Quantitativa de Sistemas de Pontuação

Abstract

Vulnerability prioritisation is a critical task in cybersecurity, particularly given the large number of vulnerabilities that organisations must evaluate and mitigate. Although traditional vulnerability scoring systems such as the Common Vulnerability Scoring System (CVSS) are widely used to support vulnerability management processes, they often fail to reflect the actual risk that vulnerabilities pose within the specific context of each organisation and to account for temporal factors, such as the evolving availability of exploits. Among these approaches, SecScore has been proposed as an enhancement to CVSS, integrating statistical models grounded in empirical evidence about the real availability of exploit code. By adjusting CVSS Threat metrics through a data-driven and explainable methodology, SecScore captures more accurately and immediately the dynamics of exploit development. However, it remains an open question whether systems such as SecScore enable more effective vulnerability prioritisation than CVSS. This work seeks to address this question by introducing an evaluation methodology that allows direct comparisons between vulnerability scoring systems. The proposed methodology is based on distance metrics relative to an ideal ordering. As a case study, the performance of SecScore in vulnerability prioritisation is evaluated and compared with that of CVSS and Exploit Prediction Scoring System (EPSS). In addition, by applying the proposed methodology, this study also examines the impact of different temporal windows on the statistics used to develop scoring systems such as SecScore.

Keywords: Vulnerability Management Process, Time-Sensitive Risk Assessment, Risk Prioritisation, Exploit Code Maturity, Quantitative Assessment of Scoring Systems

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Vulnerability management (VM) is a fundamental element of cybersecurity, encompassing the identification, assessment, prioritisation, and remediation of security vulnerabilities to help organisations reduce the likelihood of attacks to their infrastructure [37, 44]. However, VM has been a constant hurdle for defenders due to the increasing number of new vulnerabilities over the last years [43]. Just in 2024, the number of new common vulnerabilities and exposures (CVEs) surged 30% year-on-year from 17,114 to 22,254 according to Qualys researchers [36]. Thus, vulnerability prioritisation becomes an essential task in the process of vulnerability management.

In vulnerability prioritisation, the main goal is to prioritise vulnerabilities, based on how severe the damage can be to the organisation, creating a list of prioritised vulnerabilities to manage and mitigate. Nevertheless, performing this task on a regular basis can be quite challenging, not only because of the large quantity of vulnerabilities, but also because of how hard it is to effectively prioritise them. The time needed and the costs of patching can easily compromise this process of what should be patched first [6].

The Common Vulnerability Score System (CVSS) is the most common scoring system used to measure the severity of a vulnerability and is maintained by the Forum of Incident Response Teams (FIRST) [27]. CVSS is composed of three metric groups—Base, Threat, and Environmental—each with its own set of metrics and a scoring range from 0 to 10, where higher values indicate more severe vulnerabilities. This numerical score is typically translated into qualitative ratings to simplify understanding and prioritisation. The standard qualitative rating ranges from Low to Critical. However, CVSS often falls short in reflecting the true risks posed by vulnerabilities, as it does not adequately capture either the specific organisational context or the temporal dynamics of evolving threats [15]. While the Threat metrics group is intended to refine the Base score by accounting for factors that evolve over time, such as the availability of exploit code, it often lags behind real-time events, as their values depend on updates that may not keep pace with rapidly changing threat landscapes.

Additionally, the metrics are rarely used or updated by organizations due to their complexity and the manual effort required to track changes in threat intelligence feeds and exploit availability. Instead, many security teams default to using static Base scores, which may lead to a misaligned sense of prioritisation. Furthermore, the lack of automation and integration with operational data

limits the relevance of the Threat metrics in practical workflows. Due to the static nature and lack of contextual sensitivity of CVSS metrics, their effectiveness for vulnerability prioritisation is often questioned.

SecScore, an empirical scoring system that enhances the CVSS Threat metric group, addresses one of the overall weaknesses of CVSS, which is the lack of dynamic threat awareness, by incorporating empirical evidence of exploit code development [35]. In addition to being transparent in its modelling and adaptable to multiple CVSS versions, the adaptation of SecScore score over time enables better prioritisation and, consequently, better focus on critical threats. However, it remains uncertain whether a system would prioritise vulnerabilities more effectively using SecScore instead of CVSS.

Another relevant approach to vulnerability prioritisation is the Exploit Prediction Scoring System (EPSS) [30], also developed by FIRST. Unlike CVSS, which focuses on static severity metrics, EPSS predicts the likelihood that a vulnerability will be exploited in the wild within a given time window. It is a data-driven, machine learning model that combines thousands of features from multiple sources and leverages community-driven insights. Over its iterations, EPSS has evolved from a simple logistic regression model to a more sophisticated gradient-boosted decision tree implementation with over 1,400 variables, improving predictive performance and aligning with organisational remediation cycles. Given its emphasis on real-world exploitation probability, EPSS provides a useful comparison point for SecScore, which also seeks to improve prioritisation by incorporating the dynamics of exploit emergence.

Despite the continuous refinement of vulnerability scoring systems such as CVSS, SecScore, and EPSS, it remains difficult to objectively assess how much these approaches improve vulnerability prioritisation in practice. While many scoring systems demonstrate improved theoretical properties or predictive accuracy, their actual contribution to producing more effective prioritisation orders is often evaluated using heterogeneous criteria, making direct comparisons challenging. Moreover, differences in underlying assumptions, scoring objectives, and temporal dynamics further complicate the assessment of whether one approach meaningfully outperforms another. As a result, there is a lack of systematic methodologies that enable consistent and quantitative comparisons between vulnerability scoring systems with respect to prioritisation effectiveness.

1.1 Objectives

This work aims to develop an evaluation methodology that enables direct comparisons between vulnerability scoring systems based on distance metrics with respect to an ideal ordering. Using this methodology, we conduct a comprehensive evaluation of SecScore and its effectiveness in enhancing vulnerability prioritisation, with a focus on analysing how prioritisation evolves over time and identifying potential improvements to SecScore outcomes. The evaluation demonstrates how the SecScore approach significantly enhances the vulnerability prioritisation process in comparison to CVSS and the Exploit Prediction Scoring System (EPSS) [30]. Furthermore, the methodology is used to analyse the impact of different time windows on the statistics employed in the

development of scoring systems such as SecScore, and to illustrate how CVSS supports the adaptation of temporal metrics, aligning with the goal of refining prioritisation in response to evolving threat dynamics.

1.2 Document Organisation

Regarding the organisation of this document, Chapter 2 will present all the core concepts to better understand the problem that is being tackled and concepts relevant to this work. In Chapter 3, it details some relevant work in this area, where different solutions and methodologies are proposed to solve this problem. Chapter 4 provides detailed information about the problem that was tackled, as well as the methodology that was developed to approach it. Chapter 5 shows the results obtained and the multiple evaluations that were made. Finally, Chapter 6 concludes the report by explaining the results obtained and what is expected to be done in the future.

Chapter 2

Background

This Chapter provides detailed information about the concepts necessary to follow the remainder chapters of this thesis. Section 2.1 introduces Vulnerability Management Processes (VMP) and their main stages, Section 2.2 describes the different versions of the Common Vulnerability Scoring System (CVSS), including its qualitative severity ratings, and Section 2.3 presents the concept of Euclidean distance.

2.1 Vulnerability Management Processes

Vulnerability Management Processes are a systematic approach that consists of identifying, evaluating, treating, and reporting vulnerabilities, with the objective of improving the overall security posture of organisations and reducing the risk of any of these vulnerabilities being exploited [2, 5, 19]. Typically, VMPs are composed of four to six stages [2, 5, 12, 19, 32]. These approaches vary in the number and naming of stages, the automation of the discovery, the order of assessment and prioritisation, the position of the report stage, and whether there is a verification/validation stage [35]. Four-stage VMPs [12, 32] provide a basic structure but omit several elements found in more detailed models. These omissions include accounting for the combined effect of addressing multiple vulnerabilities, verifying that risk has been reduced, considering known exploits during evaluation, and performing ongoing identification of new vulnerabilities. Five-stage VMPs [2, 19] add steps for verifying that vulnerabilities have been addressed and for applying a defined method of prioritisation. Six-stage VMPs [5] include ongoing risk assessment and automated discovery. They still face issues related to tracking vulnerabilities for which no patches exist and applying prioritisation methods effectively.

The stages are usually: Discovery, Assessment/Prioritisation, Remediation, Verification/Validation, and Report. These stages are often conceptualised as part of a broader vulnerability management lifecycle in which the process operates as a continuous cycle rather than a one-time sequence of steps [20, 13]. This cyclical view emphasises that vulnerability management must be performed continuously as new vulnerabilities emerge and the organisational environment evolves.

The *Discovery* stage [2, 5, 19] starts by identifying all assets within the organisation and subsequently performing vulnerability scans to detect vulnerabilities in these assets. In this stage,

security teams typically use asset discovery tools [32], such as NMAP and Nessus [22, 41], to automate this process and maintain an up-to-date inventory of their assets. Additionally, this phase plays a crucial role in identifying vulnerabilities within the assets already listed in the inventory. The *Assessment/Prioritisation* stage will rank the vulnerabilities previously identified in the Discovery stage based on the risk they pose to the organisation in order to create a list of what should be addressed first. Commonly used resources in this stage include the Common Vulnerability Scoring System (CVSS), which provides a standardised way of scoring the severity of vulnerabilities, and the Common Vulnerabilities and Exposures (CVE) list maintained by MITRE [7], which catalogues publicly known vulnerabilities [32]. These resources support vulnerability prioritisation methods by providing severity scores and identifiers; however, they fail to account for the intricate interdependencies and contextual factors that influence the actual risk posed by vulnerabilities in real-world environments. In the *Remediation* stage, vulnerabilities are mitigated based on their severity, progressing from the most critical to the least. Vulnerabilities should be thoroughly addressed by applying security patches and correcting misconfigurations to prevent exploitation [12]. The *Verification/Validation* stage will confirm whether the vulnerabilities have been effectively addressed by performing penetration tests or rescans of the system [2, 5, 19]. Finally, the *Report* stage provides insights and documentation on the prioritised vulnerabilities identified and mitigated throughout the entire process [5, 12, 19].

2.2 Common Vulnerability Scoring System (CVSS)

In this section, we first present CVSS v2, as it is the first iteration of the system. We then provide an overview of CVSS v3.1, given its widespread adoption and continued relevance in practice. Finally, we also describe CVSS v4.0, which, although it is not yet as widely used, represents the most recent version of the framework.

2.2.1 CVSS v2

CVSS supports VMP, especially in the stage of assessment and prioritisation. It outputs numerical scores that range from 0 to 10, which are used to quantify the severity of each vulnerability that is identified [26].

As shown in Figure 2.1, CVSS v2 consists of three metric groups: Base, Temporal, and Environmental, each of which has its own set of metrics. The Base metrics must be assigned by the analyst to compute the score, whereas the Temporal metrics can be used to refine the Base score by accounting for factors such as the availability of exploit code or the level of remediation effort. Additionally, the Environmental metrics can be applied to further tailor the score to better reflect the potential impact of the vulnerability within a specific user's environment.

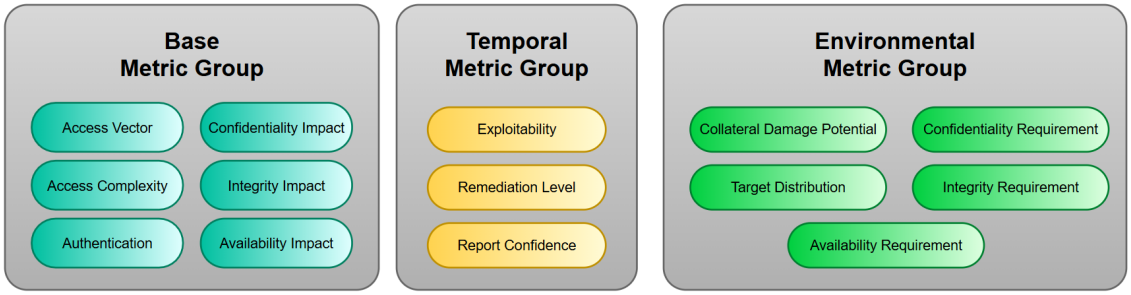


Figure 2.1: CVSS v2 Metrics Group

2.2.1.1 Base Metric Group

The Base score signifies the essential characteristics of a vulnerability that do not change over time or with the user environment. This group consists of the Access Vector (*AV*), Access Complexity (*AC*), and Authentication (*Au*) metrics, which describe how a vulnerability can be reached and whether specific conditions must be met to exploit it. It also includes three impact metrics — Confidentiality (*C*), Integrity (*I*), and Availability (*A*) — which quantify the effects of a successful exploit in each of these areas.

The *BaseScore* (Equation 2.1) combines both *Impact* (Equation 2.2) and *Exploitability* (Equation 2.3), and then adjusts the result using the function *f(Impact)* (Equation 2.4) to provide the overall CVSS v2 Base Score. The *Impact* and *Exploitability* values can be obtained using the values in Table 2.1 and Table 2.2, respectively.

Metric	Metric Value	Numerical Value
Confidentiality Impact (*C*)	Complete (*C*) Partial (*P*) None (*N*)	0.660 0.275 0
Integrity Impact (*I*)	Complete (*C*) Partial (*P*) None (*N*)	0.660 0.275 0
Availability Impact (*A*)	Complete (*C*) Partial (*P*) None (*N*)	0.660 0.275 0

Table 2.1: CVSS v2 Metric Values for Impact Metrics

$$BaseScore = \lceil ((0.6 \times Impact + 0.4 \times Exploitability - 1.5) \times f(impact)) \rceil \tag{2.1}$$

$$Impact = 10.41\,(1 - (1 - C) \times (1 - I) \times (1 - A)) \tag{2.2}$$

Metric	Metric Value	Numerical Value
Access Vector (*AV*)	Local (*L*)	0.395
	Adjacent (*A*)	0.646
	Network (*N*)	1.0
Access Complexity (*AC*)	High (*H*)	0.35
	Medium (*M*)	0.61
	Low (*L*)	0.71
Authentication (*Au*)	Multiple Instances (*M*)	0.45
	Single Instance (*S*)	0.56
	No Authentication (*N*)	0.704

Table 2.2: CVSS v2 Metric Values for Exploitability Metrics

$$Exploitability = 20 \times AV \times AC \times Au \tag{2.3}$$

$$f(impact) = \begin{cases} 0, & \text{if } Impact = 0, \\ 1.176, & \text{otherwise.} \end{cases} \tag{2.4}$$

2.2.1.2 Temporal Metric Group

The Temporal Score reflects the traits of a vulnerability that are subject to variation over time and can be calculated using Equation (2.5) and the values in Table 2.3.

Metric	Metric Value	Numerical Value
Exploitability (*E*)	Not Defined (*ND*)	1.00
	High (*H*)	1.00
	Functional (*F*)	0.95
	Proof Of Concept (*POC*)	0.90
	Unproven (*U*)	0.85
Remediation Level (*RL*)	Not Defined (*ND*)	1.00
	Unavailable (*U*)	1.00
	Workaround (*W*)	0.95
	Temporal Fix (*T*)	0.90
	Official Fix (*O*)	0.87
Report Confidence (*RC*)	Not Defined (*ND*)	1.00
	Confirmed (*C*)	1.00
	Uncorroborated (*UR*)	0.96
	Unconfirmed (*UC*)	0.95

Table 2.3: CVSS v2 Metric Values for Temporal Metrics

$$TemporalScore = \lceil BaseScore \times E \times RL \times RC \rceil \tag{2.5}$$

This metric group consists of three metrics, wherein Exploitability (*E*) reflects whether exploit methods or exploit codes are currently available. When simple, publicly accessible code exists, more individuals are able to attempt an attack, which results in a higher severity rating for the vulnerability [26]. Remediation Level (*RL*) measures how far along a vulnerability is in the process of being fixed, and Report Confidence (*RC*) expresses the degree of confidence in the existence of a vulnerability and the credibility of the technical details available about it.

It is important to note that the *TemporalScore* is not a function of time; therefore, it only changes the *BaseScore* when the evaluation is performed.

2.2.1.3 Environmental Metric Group

The Environmental score represents the aspects of a vulnerability that are pertinent to a specific user's environment. Its formula depends on two formulas: the *AdjustedImpact* (Equation 2.9), which can be obtained from the user's requirements of Confidentiality (*CR*), Integrity (*IR*), and Availability (*AR*), using the values in Table 2.4, and *AdjustedTemporal* (Equation 2.8), which is a recomputation of *TemporalScore*, where *BaseScore* is replaced by *AdjustedBaseScore* (Equation 2.7).

Metric	Metric Value	Numerical Value
Collateral Damage Potential (*CDP*)	None (*N*)	0
	Low (*L*)	0.1
	Low-Medium (*LM*)	0.3
	Medium-High (*MH*)	0.4
	High (*H*)	0.5
	Not Defined (*ND*)	0
Target Distribution (*TD*)	None (*N*)	0
	Low (*L*)	0.25
	Medium (*M*)	0.75
	High (*H*)	1.00
	Not Defined (*ND*)	1.00
Confidentiality Requirement (*CR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50
Integrity Requirement (*IR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50
Availability Requirement (*AR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50

Table 2.4: CVSS v2 Metric Values for Environmental Metrics

This metric group also consists of two additional metrics: Collateral Damage Potential (*CDP*), which describes the possibility that a vulnerability could lead to the loss or damage of physical property or equipment, and Target Distribution (*TD*), which indicates what share of systems are affected by the vulnerability. Finally, *EnvironmentalScore* can be determined using Equation (2.6).

$$EnvironmentalScore = \lceil((AdjustedTemporal+(10-AdjustedTemporal)\times CDP)\times TD)\rceil \tag{2.6}$$

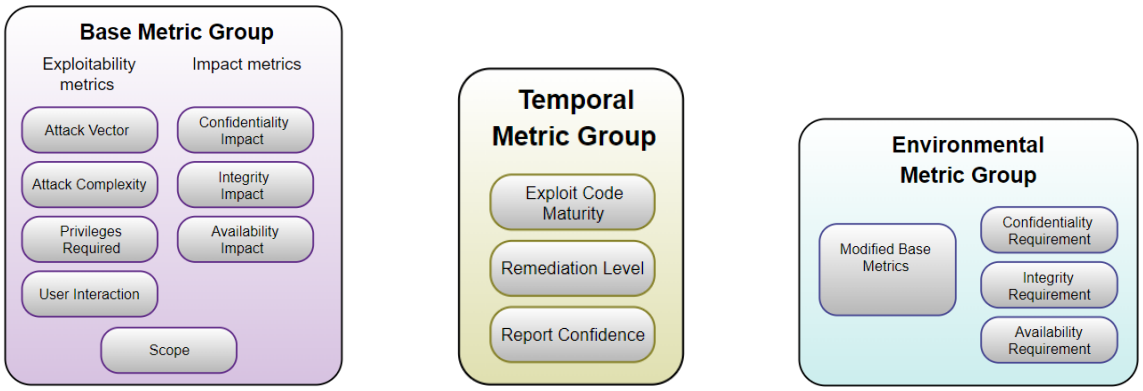
$$AdjustedBaseScore = \lceil((0.6\times AdjustedImpact+0.4\times Exploitability-1.5)\times f(impact))\rceil \tag{2.7}$$

$$AdjustedTemporal = \lceil(AdjustedBaseScore \times Exploitability \times RL \times RC)\rceil \tag{2.8}$$

$$AdjustedImpact = \min[10, 10.41\times(1-(1-CI\times CR)\times(1-II\times IR)\times(1-AI\times AR))] \tag{2.9}$$

2.2.2 CVSS v3.1

Despite the release of CVSS v4.0 in November 2023, CVSS v3.1 remains widely used, primarily due to its extensive integration into existing systems. This version of CVSS provides multiple updates compared to its previous version, and Figure 2.2 already reflects the profound changes in the three metric groups. Therefore, this section provides details about CVSS version 3.1.



(a) Base Group (b) Temporal Group (c) Environmental Group

Figure 2.2: CVSS v3.1 Metrics Group [28]

2.2.2.1 Base Metric Group

The Base Metric Group now consists of two sets of metrics: Exploitability metrics and Impact metrics. Figure 2.2a shows that Exploitability metrics are composed of four metrics: Attack Vector (*AV*), Attack Complexity (*AC*), Privilege Required (*PR*), and User Interaction (*UI*), while the

Impact metrics consist of three metrics, namely Confidentiality Impact (*C*), Integrity Impact (*I*), and Availability Impact (*A*).

The *BaseScore* equation derives from the sum of *Impact* (*I*) and *Exploitability* (*X*) and depends on sub-formulas for Impact Sub-Score (*ISS*) (Equation 2.13), which can be obtained from the metrics in Table 2.5, *Impact* (Equation 2.14), and *Exploitability* (Equation 2.15), which can be obtained from the metrics in Table 2.6. *Exploitability* indicates the ease with which a vulnerability can be exploited, while *Impact* represents the direct consequences of a successful exploit, reflecting the effects on the entity that experiences them. The *BaseScore* will be 0 if *Impact* is equal to or less than 0, and its equation will vary depending on whether the Scope (*S*) was changed (Equation 2.12) or unchanged (Equation 2.11).

Metric	Metric Value	Numerical Value
Confidentiality Impact (*C*)	High (*H*)	0.56
	Low (*L*)	0.22
	None (*N*)	0
Integrity Impact (*I*)	High (*H*)	0.56
	Low (*L*)	0.22
	None (*N*)	0
Availability Impact (*A*)	High (*H*)	0.56
	Low (*L*)	0.22
	None (*N*)	0

Table 2.5: CVSS v3.1 Metric Values for Impact Metrics

Metric	Metric Value	Numerical Value
Attack Vector (*AV*)	Network (*N*)	0.85
	Adjacent (*A*)	0.62
	Local (*L*)	0.55
	Physical (*P*)	0.20
Attack Complexity (*AC*)	Low (*L*)	0.77
	High (*H*)	0.44
Privilege Required (*PR*)	None (*N*)	0.85
	Low (*L*)	0.62 or 0.68[1]
	High (*H*)	0.27 or 0.5[1]
User Interaction (*UI*)	None (*N*)	0.85
	Required (*R*)	0.62

Table 2.6: CVSS v3.1 Metric Values for Exploitability Metrics

$$BaseScore = 0 \quad \text{if } I \leq 0 \tag{2.10}$$

$$BaseScore = \lceil \min[(I + X), 10] \rceil \quad \text{if } S = \text{Unchanged} \tag{2.11}$$

[1] If Scope (S) is changed.

$$BaseScore = \lceil \min[1.08 \times (I + X), 10] \rceil \quad \text{if } S = \text{Changed} \tag{2.12}$$

$$ISS = 1 - [(1 - C) \times (1 - I) \times (1 - A)] \tag{2.13}$$

$$I = \begin{cases} 6.42 \times ISS & \text{if } S = \text{Unchanged} \\ 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} & \text{if } S = \text{Changed} \end{cases} \tag{2.14}$$

$$X = 8.22 \times AV \times AC \times PR \times UI \tag{2.15}$$

2.2.2.2 Temporal Metric Group

The Temporal Score still reflects the traits of a vulnerability that are subject to variation over time and continues to use Equation (2.5) to calculate its score, now using the values in Table 2.7. The real change in this group is the replacement of the metric Exploitability with Exploit Code Maturity (*E*), while the other two remain the same.

Metric	Metric Value	Numerical Value
Exploit Code Maturity (*E*)	Not Defined (*X*)	1.00
	High (*H*)	1.00
	Functional (*F*)	0.97
	Proof Of Concept (*P*)	0.94
	Unproven (*U*)	0.91
Remediation Level (*RL*)	Not Defined (*X*)	1.00
	Unavailable (*U*)	1.00
	Workaround (*W*)	0.97
	Temporal Fix (*T*)	0.96
	Official Fix (*O*)	0.95
Report Confidence (*RC*)	Not Defined (*X*)	1.00
	Confirmed (*C*)	1.00
	Reasonable (*R*)	0.96
	Unknown (*U*)	0.95

Table 2.7: CVSS v3.1 Metric Values for Temporal Metrics

Exploit Code Maturity (*E*) now measures the likelihood of a vulnerability being attacked, and it is based on the current state of exploit techniques, exploit code availability, or active, "in-the-wild" exploitation [28]. When readily usable exploit code is publicly available, individuals without specialised skills can attempt an attack, which leads to a higher severity rating for the vulnerability.

2.2.2.3 Environmental Metric Group

The Environmental Metric Group was also changed, and its score now depends on three sub-formulas: the Modified Impact Sub-Score (*MISS*) (Equation 2.16), which can be obtained from the

Modified Impact Metrics Confidentiality (*MC*), Integrity (*MI*), and Availability (*MA*), while also considering the user's requirements of Confidentiality (*CR*), Integrity (*IR*), and Availability (*AR*), using the values in Table 2.8; Modified Exploitability (*ME*) (Equation 2.19), which derives from the Modified Exploitability Metrics Attack Vector (*MAV*), Attack Complexity (*MAC*), Privilege Required (*MPR*), and User Interaction (*MUI*); and Modified Impact (*MI*), which is calculated in a similar way to Equation (2.14) in Equations (2.17) and (2.18). The organisation of these components within the Environmental metrics group is illustrated in Figure 2.2c. Finally, *EnvironmentalScore* can be determined with Equation (2.21) if the scope (*S*) is unchanged and (2.22) if the scope changes. The environmental score is 0 if *MI* is less than or equal to 0 (Equation 2.20).

Metric	Metric Value	Numerical Value
Confidentiality Requirement (*CR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50
Integrity Requirement (*IR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50
Availability Requirement (*AR*)	Not Defined (*X*)	1.00
	High (*H*)	1.50
	Medium (*M*)	1.00
	Low (*L*)	0.50

Table 2.8: CVSS v3.1 Metric Values for Environmental Metrics

$$MISS = \min[1 - ((1 - CR \times MC) \times (1 - IR \times MI) \times (1 - AR \times MA)), 0.915] \quad (2.16)$$

$$ModifiedImpact = 6.42 \times MISS \quad \text{if } S = \text{Unchanged} \quad (2.17)$$

$$ModifiedImpact = 7.52 \times (MISS - 0.029) - 3.25 \times (MISS \times 0.9731 - 0.02)^{13} \quad (2.18)$$

$$\text{if } S = \text{Changed}$$

$$ModifiedExploitability = 8.22 \times MAV \times MAC \times MPR \times MUI \quad (2.19)$$

$$ES = 0 \quad \text{if } MI \leq 0 \quad (2.20)$$

$$ES = \lceil \lceil \min[(MI + MX), 10] \rceil \times E \times RL \times RC \rceil \quad \text{if } S = \text{Unchanged} \quad (2.21)$$

$$ES = \lceil \lceil \min[1.08 \times (MI + MX), 10] \rceil \times E \times RL \times RC \rceil \quad \text{if } S = \text{Changed} \tag{2.22}$$

2.2.3 CVSS v4.0

The Common Vulnerability Scoring System v4.0 introduces key updates to improve the measurement and communication of software vulnerability risks. [3, 9].

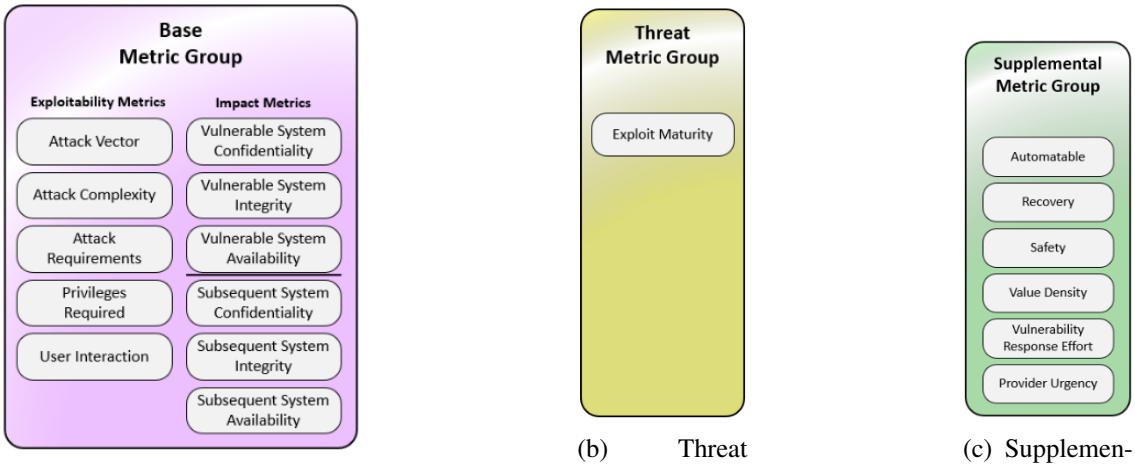
The first change comes with the nomenclature of the scores due to the popularity of the CVSS Base Score, the numerical value that evolved to mean the same as the overall CVSS score [3, 31]. Since numerical CVSS scores have very different meanings, the score should now be labeled based on the metrics used to calculate them as shown in Table 2.9 [3, 29].

Nomenclature	CVSS Metrics Used
CVSS-B	Base
CVSS-BE	Base and Environmental
CVSS-BT	Base and Threat
CVSS-BTE	Base, Threat, Environmental

Table 2.9: CVSS Nomenclatures

CVSS v4.0 now consists of four metric groups: Base, Threat, Environmental and Supplemental, each having its own set of metrics, as it is illustrated in Figure 2.3.

In the Base Metric Group (Figure 2.3a), a new metric was added under the name of Attack Requirement (*AT*). This metric evaluates the prerequisite conditions or variables inherent to the deployment and execution of a vulnerable system that facilitate an attack. The User Interaction (*UI*) base metric was updated, in order to increase its granularity.



(a) Base Group

(b) Threat Group

(c) Supplemental Group

Figure 2.3: CVSS v4.0 Modified Metrics Group [29]

Due to its lack of clarity and inconsistencies that were brought to the score, Scope (*S*) metric was also retired. However, to address the impact on both vulnerable and affected systems, the Impact metrics were divided in two sets: Vulnerable System Impact metrics and Subsequent System

Impact metrics. The first set covers the confidentiality (*VC*), integrity (*VI*), and availability (*VA*) of the vulnerable system while the second set comprises the confidentiality (*SC*), integrity (*SI*), and availability (*SA*) of subsequent systems, referring to systems that become vulnerable due to the consequences of the initial exploit.

Metric Name	Description
Safety (*S*)	Degree of impact to the Safety of a human participant that can be injured as a result of exploitation
Automatable (*AU*)	Provides information about the possibility of automate exploitation
Provider Urgency (*U*)	Rating provided by the vendor for this vulnerability
Recovery (*R*)	Ability of a system to recover services after the attack
Value Density (*V*)	Amount of resources that the attacker will gain control because of a single exploitation event
Vulnerability Response Effort (*RE*)	Amount of effort required to respond to the impact of the vulnerabilities

Table 2.10: CVSS v4.0 Supplemental Metrics

The Threat Metric Group (Figure 2.3b) represents the old Temporal Metric Group. Besides the renaming of the metric group, the metrics Remediation Level (*RL*) and Report Confidence (*RC*) were retired with the purpose of reducing the CVSS-BTE. The metric Exploit Code Maturity was also renamed to just Exploit Maturity and got the metric values High (*H*) and Functional (*F*) merged into one value called Attacked (*A*).

This version of CVSS added a new optional metrics group called Supplemental (Figure 2.3c) that provides additional extrinsic characteristics of a vulnerability and context. These can be seen in Table 2.10. These metrics do not have any impact on the calculated score, as they only serve to better understand the extrinsic information about vulnerabilities.

2.2.4 Qualitative Severity Ratings

CVSS provides a qualitative severity rating that enables the categorization of vulnerabilities based on their CVSS score. These ratings correspond to numerical ranges, as can be seen in Table 2.11, allowing a better understanding of the vulnerability severity, and it also helps in prioritisation response efforts [8, 28, 29].

A vulnerability with a score of 0 means that it has no impact or poses no risk. Usually, it is used for non-exploitable vulnerabilities. If the score falls between the values of 0.1 and 3.9, it

CVSS Score Range	Qualitative Rating
0	None
0.1 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 8.9	High
9.0 - 10.0	Critical

Table 2.11: CVSS Qualitative Severity Ratings

will be categorised as Low, meaning that these vulnerabilities pose minimal risk and have minimal impact, requiring unlikely circumstances for exploitation. If a vulnerability has a score of 4.0 to 6.9, it will have a severity rating of Medium, which means that remediation is not urgent, but the exploitation of these vulnerabilities is possible. Even though they might not present significant damage, security teams must keep an eye on them.

A score of 7.0 to 8.9 represents a High severity vulnerability, meaning that its exploitation has a serious impact and calls for swift resolution. Finally, a score of 9.0 to 10.0 stands for Critical severity. Vulnerabilities that fall in this range should be mitigated immediately, as their exploitation can cause catastrophic consequences.

2.3 Euclidean Distance

The Euclidean distance is used to measure the straight-line distance between two points in a geometric space [38]. In this work, the Euclidean distance is used to objectively quantify and compare differences between SecScore and CVSS prioritisation lists, enabling meaningful analysis and interpretation. By giving two points, p and q, in n-dimensional Euclidean space, the distance d between these points can be calculated as follows [38]:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.23}$$

This metric is often used as a distance metric in mathematics, machine learning, and statistics, as it gives a perception of the spatial relationship between points in any dimensional space. Further explanations about its use in this work will be given in Section 4.3.

Chapter 3

Related Work

This Chapter provides a comprehensive overview of the state of the art in vulnerability assessment and prioritisation, with a particular focus on approaches that extend or complement CVSS. It is organised to progressively move from automation and learning-based enhancements of CVSS metrics (Section 3.1), to approaches that incorporate temporal dynamics into vulnerability scoring (Section 3.2). Section 3.3 discusses known inconsistencies and critiques of CVSS, establishing the limitations that motivate alternative scoring approaches. SecScore and EPSS are presented in dedicated sections—Sections 3.4 and 3.5, respectively—providing detailed descriptions of their methodologies and design choices, as both scoring systems are central to the comparative evaluation conducted in later chapters. Finally, broader vulnerability evaluation methodologies are examined in Section 3.6, followed by an overview of distance-based techniques used for vulnerability comparison and analysis in Section 3.7.

3.1 Automation and Machine Learning for CVSS Metrics

By leveraging artificial intelligence (AI) and machine learning (ML) techniques, traditional frameworks like CVSS can be enhanced to provide more accurate, dynamic, and context-aware assessments of vulnerabilities. Zhen *et al.* [43] presents ILLATION, a system that uses artificial neural networks (ANN), natural language processing (NLP) and probabilistic logic programming to learn the adversary behaviour and network interactions, in order to improve vulnerability risk prioritisation. ILLATION learns from past data to predict the risk of new vulnerabilities and adjusts initial risk scores based on network changes. The authors proved that this system can handle large-scale vulnerability assessments, and it outperformed CVSS in adversary-specific scenarios. However, ILLATION focuses primarily on host reachability and does not fully consider layer-2 attacks or the multi-hop exploitation chains in-depth. Additionally, it relies on certain assumptions regarding network defense, such as no defense rule conflicts (e.g., no overlapping firewall rules) and limited focus on direct vulnerability exploitation.

Zhun *et al.* [44] similarly to ILLATION, presents LICALITY, a system that prioritises vulnerabilities by evaluating the likelihood and criticality of exploitation through deep learning and logical reasoning. It uses latent features and exploit records to determine the likelihood of ex-

ploitation and assesses the impact of successful exploitation with CVSS metrics. The authors evaluated LICALITY with two real-world case studies, where they show the improvements in prioritisation and in remediation workload when compared to methods that only use CVSS. Still, the approach assumes that attackers exhibit consistent preferences based on previous exploits and system features. If the attacker behaviour deviates significantly, the model assumptions may not hold. Additionally, the performance of the system depends on the alignment between historical and future threats. If the characteristics of a future threat differ from historical records, the prioritisation accuracy may decrease.

On the other hand, Virender [10] also uses deep learning to overcome the limitations of rule-based systems and static metrics like CVSS, by developing a framework capable of identifying and addressing vulnerabilities dynamically. The author specifically uses Convolutional Neural Networks (CNN) with three convolutional layers, max-pooling layers, and attention mechanisms to improve prioritisation and remediation. This study demonstrated that this system outperforms CVSS across all metrics and has a high capability in distinguishing high-risk vulnerabilities with very few false positives and negatives. However, deep learning models—particularly those incorporating attention mechanisms—can be highly resource-intensive, often requiring substantial computational capacity for both training and inference. Moreover, instances of misalignment were observed between the model's recommendations and expert assessments, highlighting the need for further refinement in the underlying decision-making logic.

Shaofeng *et al.* [16] proposes a ML-based method that aims to reduce scoring errors and enhance vulnerability prioritisation. The authors use algorithms to identify and remove redundant metrics and reduce the metrics with low contribution to the overall score or with high correlation, then they implement a decision tree to construct a simple automated evaluation model and finally refine the scoring by combining classification and correlation analysis. The implementation of this system improved accuracy by reducing scoring inconsistencies from 20% to 40% and showed that the scoring errors decreased significantly when metrics such as Scope and Attack Complexity were removed. In addition, it enhanced efficiency by reducing the processing time by 72%. However, removing metrics such as Scope and Attack Complexity might omit critical contextual factors, potentially affecting the assessment of more complex vulnerabilities.

Another approach is taken by Andrey *et al.* [25] where the authors designed a system to enhance the assessment of software and hardware vulnerabilities using text mining and NLP. This system leverages Doc2Vec to create text feature vectors and uses NN and regression models to predict CVSS metric components and scores. The proposed method showed improvements for vulnerability assessment in terms of speed and accuracy and overall improvement for threat assessment efficiency. Nevertheless, the system heavily relies on the NVD database for textual vulnerability descriptions and, as noted in the analysis made by the authors, NVD has issues with data inconsistencies and incomplete entries, which might affect the accuracy of predictions.

3.2 Temporal Scoring in CVSS

As it was previously mentioned, both versions of CVSS incorporate a temporal and environmental score. These are not time-dependent, as they only change when experts do a re-evaluation. Consequently, they fail to capture the evolution of threats over time, as they are not dynamically estimated.

Ensar and Weizhi [37] introduced XVRS (Extended Vulnerability Risk Scoring), which is a new algorithm that integrates real-time threat intelligence to improve temporal scoring. It incorporates inputs from multiple data sources, such as social media, vulnerability databases, and public code repositories, and assigns points based on mentions on social media or the presence of exploit code in repositories. By combining CVSS and these external metrics, the algorithm dynamically changes the score.

On the other hand, Artur *et al.* [4] presented a way to automatically calculate the CVSS temporal score, but with a focus on Apache Log4j vulnerabilities from 2021, specifically on four CVEs (CVE-2021-44228, CVE-2021-45046, CVE-2021-45105, CVE-2021-44832). The authors developed a Python application that collects data from public data sources and performs an analysis to assess the exploitation stages and response measures. The application was able to calculate the temporal scores and show the evolution over time, providing better prioritisation insights for these CVEs.

Another solution is presented by Takashi and Masaya [24], where they propose and evaluate security risk growth models for assessing vulnerabilities. The authors evaluate three models—Exponential Distribution, Gamma Distribution, and Gompertz Curve—to describe the likelihood of exploitation over time. These models illustrate how risk evolves dynamically, providing a time-dependent view of vulnerability risks and shifting priorities.

3.3 Inconsistencies and Critiques of CVSS

CVSS, although extensively used as a standard to assess the severity of vulnerabilities, has been prone to criticism and demonstrates significant inconsistencies [11, 39, 42, 45]. These issues cast doubt on its usefulness in the management of vulnerabilities.

One of the principal critiques is the lack of empirical justification for its scoring. There is a lack of transparency when the weights are assigned to the metrics and the final scores are calculated, which raises concerns about the robustness of the system [11, 39]. This obscurity also extends to scoring formulas, as they remain undocumented and unexplained, despite having a key role in the functionality of the framework [39]. In addition, CVSS aims to assess the technical severity of vulnerabilities rather than risk, not taking into account important aspects such as the likelihood of exploitation or organisational impact, making it inadequate for risk-based decision-making [11, 39].

The scoring model is also affected by overgeneralisation, leading to inflated and clustered scores. Many vulnerabilities are assigned medium to high scores due to the ambiguous guidelines

that lead evaluators to assume the worst case when they are faced with uncertainty, hindering effective prioritisation [11].

Metrics such as Scope and Attack Complexity are interpreted differently by the evaluators, resulting in scoring variability across similar vulnerabilities. Identical vulnerabilities often receive disparate scores, which reduces confidence in the framework [39, 42, 45]. Also, human bias further complicates scoring, as analysts rely on personal judgement to solve ambiguities, which introduces variability and usually overestimates or underestimates severity [42].

The system also faces challenges in addressing the contextual nuances of vulnerabilities. It does not adequately capture the behaviour of vulnerabilities in shared libraries, chained exploit scenarios, or specific environments, demonstrating its inability to provide meaningful insights into domain-specific vulnerabilities [39]. Additionally, scores for the same vulnerability can change over time or between evaluators, highlighting the dynamic nature of the scoring process. Studies have shown that evaluators change the score without additional input, which shows inherent inconsistencies in CVSS [42].

3.4 SecScore

SecScore is a novel approach to vulnerability severity scoring that enhances CVSS by integrating empirical evidence of exploit code development [35]. It addresses the limitations of CVSS by integrating dynamic, time-sensitive metrics into its scoring, with the intention of improving vulnerability prioritisation in risk-based cybersecurity management processes.

The authors used the Asymmetric Laplace (AL) distribution to model the probability of exploit code availability over time, which allows SecScore to dynamically adjust scores based on the time elapsed since a vulnerability publication and the statistical likelihood of exploitation. Through the assimilation of parameters such as location, scale, and asymmetry, SecScore captures details of exploit code emergence across different vulnerability profiles [35].

To validate SecScore, the authors built a dataset that integrates information from CVE, ExploitDB, and CVEdetails, with more than 27,000 vulnerabilities. This dataset includes details on vulnerability types, platforms, and exploit publication dates. Statistical analysis showed that the AL distribution outperformed the Skewed Normal and Laplace distributions in representing the probability of exploit code development by presenting lower mean squared error (MSE) values.

Regarding SecScore evaluation, the authors highlight its ability to improve prioritisation. They demonstrate how SecScores enable distinctions between vulnerabilities with identical CVSS scores by including the time-dependent likelihood of exploitation. In addition, SecScore also proved that it can make better assessments of the actual vulnerability risk level by dynamically adjusting the scores over time. Using statistical models, the authors introduce, through SecScore, an approach to vulnerability prioritisation that is transparent and explainable. The integration of time-sensitive metrics addresses limitations in CVSS, providing more effective risk management.

While SecScore does, in fact, introduce multiple innovations over frameworks like CVSS, it is not made clear by the authors whether SecScore outperforms CVSS or even EPSS. Moreover,

while SecScore does indeed improve the prioritisation of vulnerabilities with identical CVSS scores, it is not clear if this granularity translates to meaningful prioritisation improvements in practice. In contrast, CVSS's simpler system or EPSS predictions may offer clearer and more actionable insights. Another consideration is SecScore's focus on exploit likelihood, which might deprioritise vulnerabilities with a lower probability of exploitation but severe potential impact.

3.5 Exploit Prediction Scoring System

EPSS is a vulnerability prioritisation model designed to improve existing frameworks such as CVSS [14, 15, 30]. It aims to predict the likelihood of vulnerability exploitation using a data-driven approach enriched by community-driven insights [15, 30]. The system seeks to overcome the limitations of CVSS, particularly the inability of CVSS to incorporate post-disclosure information.

The authors explain that the motivation behind EPSS arises from the increasing rate of disclosed vulnerabilities and the limited capacity of organisations to remediate them. In 2022, MITRE disclosed more than 25k vulnerabilities, but only 5% of these were exploited in the wild, showing the inefficiency of current prioritisation strategies that rely heavily on static scoring systems [15, 30]. In addition, the authors highlight that existing approaches also lack the transparency, scope, and adaptability required to meet the needs of practitioners.

Throughout the paper, the authors show how EPSS has evolved over its three iterations. The first version implemented a logistic regression model trained on 16 variables to predict the likelihood of exploitation within the first year after a vulnerability disclosure. This version outperformed CVSS in terms of precision and recall, but its simplicity and reliance on limited features impeded broader adoption.

The second version introduced a centralised architecture and a more complex model using XGBoost. This version extended the set of features to more than 1,000 variables, resulting in improved predictive performance. However, the second version still had limitations, particularly in aligning the prediction timelines with the typical 30-day remediation cycles of organisations.

Finally, the third version addresses these issues by refining the model and expanding the data sources. The system now aggregates data from multiple sources, such as Exploit-DB, GitHub, Fortinet, and AlienVault, to build a comprehensive dataset, allowing EPSS to predict the probability of exploitation activity within a 30-day window. The feature set now includes 1477 variables that cover multiple metrics. The model is implemented using gradient-boosted decision trees developed in XGBoost, and to optimise performance, extensive hyperparameter tuning and feature selection were performed. This version showed an improvement of 82% over the second version, with major gains in precision and recall.

Despite its improvements, the authors point out some limitations. The model depends heavily on external data sources, some of which are commercially controlled, restricting full reproducibility and potentially introducing bias or gaps in coverage. Its predictions rely on exploit activity detected through signature-based intrusion detection systems, which means that missed,

unmapped, or unattributed exploitation events may not be reflected in the training data. EPSS is also constrained to vulnerabilities with CVE identifiers and, therefore, excludes other exploited weaknesses, such as configuration errors or flaws without assigned CVEs [14]. The feature set, although large, may still fail to capture contextual or conditional factors that influence exploitation, such as environment-specific prerequisites or software interactions [14]. Additionally, because exploitation patterns evolve over time, the model requires periodic retraining, which may shift variable influence and complicate long-term interpretability [14, 15, 30]. The authors further note a theoretical risk that public exploitability predictions could influence adversary behaviour and clarify that EPSS estimates threat likelihood only, without incorporating asset value, defensive posture, or remediation cost [14, 15, 30].

3.6 Vulnerability Evaluation Methods

Vulnerability evaluation has been approached through multiple methodologies in the literature, ranging from lifecycle analysis to statistical modelling and alternative scoring frameworks. Different works propose distinct mechanisms for assessing severity, exploitability, and prioritisation.

Raza and Ahmed [33] conduct a systematic review of vulnerability lifecycle models used in operating systems. Their work describes how vulnerabilities progress through stages such as discovery, disclosure, exploit availability, and patch release. The authors highlight how lifecycle events are used to evaluate the urgency and severity of vulnerabilities, noting that some studies classify vulnerabilities into zero-day or potential-risk categories based on factors such as age, exploit existence, and patch readiness. The review shows that lifecycle-based evaluation helps identify when vulnerabilities are most exposed and how delays in patching relate to exploitation. Limitations arise from the descriptive nature of the review, as the authors do not propose quantitative models and rely on secondary studies whose methodologies vary widely.

Extending beyond lifecycle analysis, Koscinski *et al.* [18] perform an empirical comparison of four scoring systems—CVSS, Stakeholder-Specific Vulnerability Categorisation (SSVC), EPSS, and the Exploitability Index—using 600 vulnerabilities from Microsoft's Patch Tuesday reports. The study evaluates the consistency, triage support, and exploitability alignment of each scoring system by analysing scoring disagreements, prioritisation overlap, and time-based exploit prediction behaviour. Their findings reveal a low correlation across systems, substantial variation in severity classifications, and limited overlap in top-priority vulnerabilities. The authors also report that EPSS rarely assigns high exploitability probabilities before vulnerabilities appear in the CISA KEV catalogue. A limitation identified in the study is that the dataset consists solely of Microsoft vulnerabilities, which may not capture behaviours present in broader software ecosystems.

In a different line of work, Renney *et al.* [34] propose the V-Score, a two-layer vulnerability risk scoring system designed to address the limitations of existing scoring approaches. The authors define a global scoring layer derived from OSINT components and a local layer that incorporates contextual organisational factors. They evaluate the system through a user study and show that the V-Score achieves higher accuracy and recall than CVSS when identifying high-risk vulnera-

bilities referenced by the Computer Emergency Response Team Coordination Center (CERT/CC). Their results show that V-Score achieves 75% accuracy and 52% recall when identifying high-risk vulnerabilities, compared to 39% accuracy and 18% recall for CVSS. The authors note that one limitation is the dependence on the availability and quality of OSINT data sources, which may vary across organisations and time periods.

Another solution is presented by Liu and Zhang [21] where the authors introduce the Vulnerability Rating and Scoring System (VRSS), a combined qualitative and quantitative scoring method intended to unify disparate vulnerability rating systems. Their analysis includes over 33,000 vulnerabilities from major databases, such as IBM ISS X-Force, VUPEN, and NVD and examines statistical distributions across rating systems. VRSS integrates multiple scoring practices to generate consistent rating outputs and addresses distributional inconsistencies observed in traditional systems. In the authors experiments, VRSS produces vulnerability distributions that more closely follow expected statistical patterns, addressing inconsistencies observed in earlier rating systems. A noted limitation is that VRSS relies on handcrafted weight assignments, which may require recalibration as vulnerability landscapes evolve.

From a statistical modelling perspective, Ruchansky *et al.* [1] propose a statistical framework for prioritising vulnerabilities using mid-quantile regression and ordinal modelling. The method estimates the probability that a vulnerability falls into a higher-severity class and evaluates performance using accuracy metrics and empirical exploitation data. Their evaluation uses ranking and calibration metrics and compares model predictions against empirical exploitation events. The authors report that their approach produces well-calibrated ordinal risk estimates and improves ranking performance relative to baseline models. However, they also highlight that the reliability of the method depends on the availability of accurate exploitation labels and that performance decreases when the training data contains sparse or inconsistent exploit information.

3.7 Distance Metrics as a Method for Evaluating Vulnerabilities

Distance-based techniques have also been explored as a way to compare and analyse vulnerabilities by quantifying the similarity between metric representations or structured data. These approaches rely on mathematical distance functions to support tasks such as classification, clustering, and sequence comparison.

Kebande *et al.* [17] develop the Vulnerability Similarity Measure (VSM), which evaluates the similarity between vulnerabilities based on their CVSS metric vectors. The authors use Hamming distance and Euclidean distance to quantify differences between vulnerabilities and classify them into similarity-based groups. Their approach enables the comparative assessment of vulnerabilities by analysing the distance between their metric attributes. In this work, their experiments show that these distance measures allow for the identification of clusters of vulnerabilities that share structural CVSS properties, enabling an alternative view of prioritisation beyond static severity scores.

Building on the idea of comparing vulnerabilities through mathematical dissimilarity, Khan

et al. [23] apply clustering methods to vulnerability datasets using both quantum and classical algorithms. Classical K-means clustering, which relies on Euclidean distance for centroid assignment, is used as a baseline for evaluating the grouping of vulnerabilities by characteristics such as severity, vendor, and exploitability. The authors compare clustering performance across methods to identify structural patterns within vulnerability data. Their results show that classical Euclidean-based clustering tends to form coarse groupings aligned with broad severity classes, while quantum clustering methods capture finer structural distinctions within the data. Although Euclidean distance enables straightforward grouping, the authors note that its effectiveness depends on the choice and scaling of features. Vulnerabilities represented by sparse or categorical attributes may lead to unstable or low-quality clusters.

Further expanding the scope of distance-based analysis, Studer and Ritschard [40] provide a review of dissimilarity measures for sequences, including Hamming distance, optimal matching distance, and Euclidean-like metrics. Although not focused on cybersecurity, their analysis describes how different distance functions capture variations in state sequences, timing, duration, and ordering. Their results show that no single distance measure captures all structural aspects of sequence data, and each measure emphasises different dimensions depending on its design. A limitation identified by the authors is that some distances, such as Hamming distance, are highly sensitive to sequence position and may overlook timing or duration differences, whereas others, such as optimal matching, depend heavily on cost parameter choices. These concepts are applicable to vulnerability analysis when modelling vulnerabilities or their associated events as sequences with categorical or temporal structures.

Overall, the reviewed literature highlights a wide range of approaches for vulnerability assessment and prioritisation, including learning-based enhancements to CVSS, temporal scoring models, alternative evaluation frameworks, and distance-based analytical techniques. While these contributions address important limitations of traditional scoring systems, they also reveal a lack of methodologies for quantitatively comparing vulnerability prioritisation outcomes across different scoring approaches. In particular, existing works either focus on improving scoring accuracy or analysing vulnerability characteristics, but rarely provide a principled way to evaluate how close a prioritisation produced by a scoring system is to an ideal or reference ordering. This gap motivates the methodology introduced in the following chapter, which leverages distance metrics to enable direct, quantitative comparisons between vulnerability prioritisation strategies and to assess their effectiveness over time.

Chapter 4

Proposed solution

This Chapter presents a comprehensive analysis of the proposed improvements in vulnerability prioritisation methodologies, focusing on SecScore and enhancements to the CVSS framework. SecScore's innovations are examined to determine their efficacy compared to established frameworks such as CVSS and EPSS, with a focus on empirical validation through custom metrics and systematic evaluation.

While SecScore does, in fact, introduce multiple innovations over frameworks like CVSS, it is not made clear by the authors whether SecScore actually outperforms CVSS or even EPSS. It has been proven that it prioritises better than CVSS by being able to differentiate vulnerabilities with the same CVSS score and providing a nuanced and time-sensitive approach. However, questions remain regarding the implications of its prioritisation methodology and whether its refined scoring mechanisms consistently result in better security outcomes. While SecScore leverages statistical models to update scores based on exploit likelihood, its reliance on historical data unveils potential limitations in handling new vulnerabilities or quickly evolving threats.

4.1 Methodology for the Assessment of Scoring Systems

With the aim of conducting a thorough evaluation of the performance of scoring systems in vulnerability prioritisation, a method was developed that enables a comprehensive analysis of the effectiveness of each framework in supporting vulnerability management. As part of this approach, evaluation metrics were designed specifically to assess the effectiveness of vulnerability prioritisation and to allow for a relative comparison between different existing scoring systems.

The method, as depicted in Figure 4.1, is based, first, on a list of vulnerabilities (Input 1, Section 4.1.1) that will be used for comparison. From this list, an ideal prioritisation is defined(Input 2, Section 4.1.2), serving as the baseline or ground truth (Section 4.2). Next, the distance is calculated between the ideal prioritisation and the one obtained through a given scoring system (Section 4.3). Finally, the performance comparison among different vulnerability scoring systems is carried out based on the distance between the prioritisation generated by each system and the ideal prioritisation (Section 4.4).

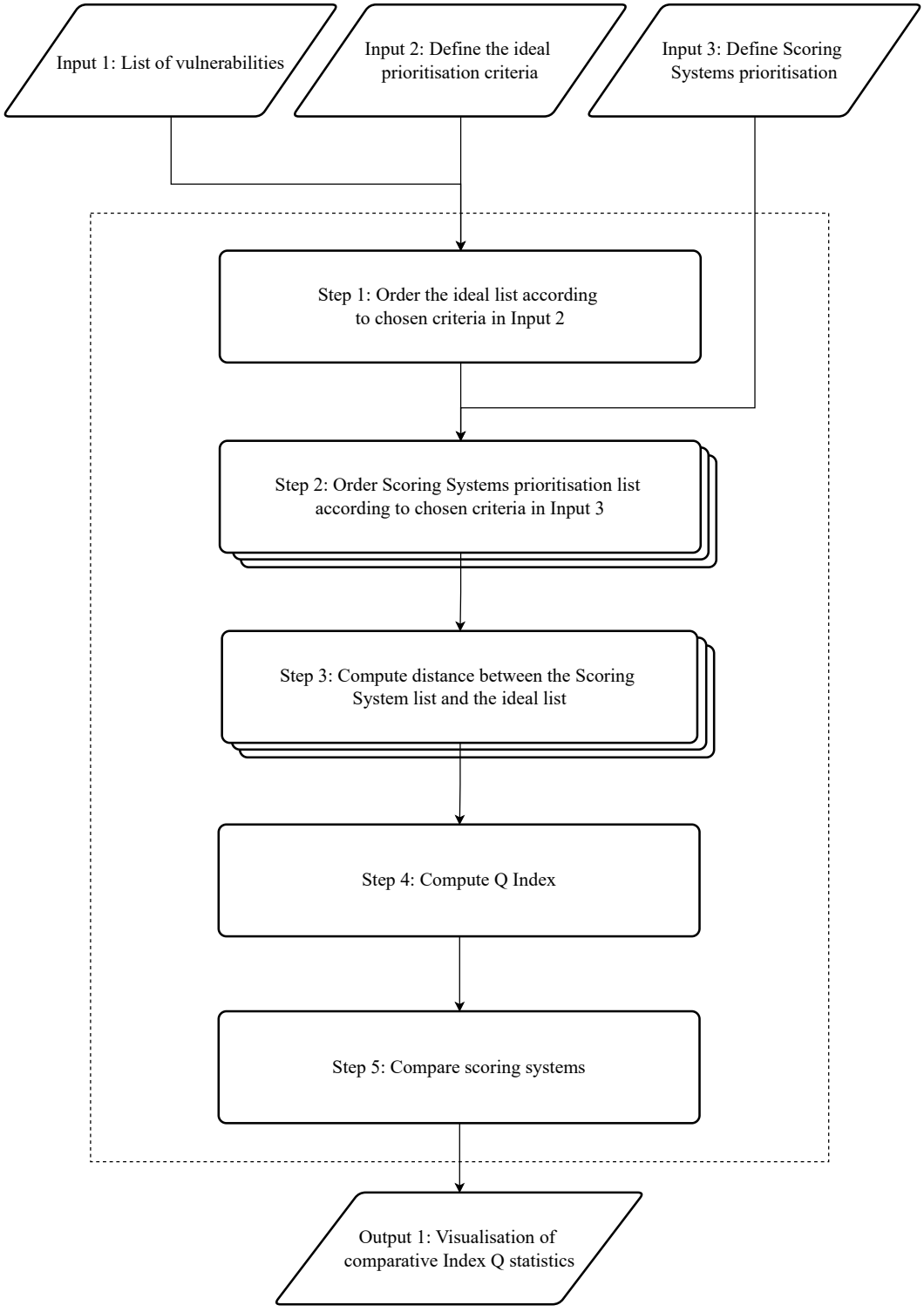


Figure 4.1: Methodology for the Assessment of Scoring Systems Flowchart

4.1.1 Input 1: List of Vulnerabilities

The methodology operates on a user-provided list of vulnerabilities, where each entry represents an individual vulnerability. No assumptions are made regarding the internal structure, format, or specific attributes of this list. Instead, the dataset is treated as an abstract collection of vulnerabilities to which different prioritisation criteria can be applied.

This list constitutes the common data input for the entire process. It is first used to generate the reference prioritisation and is subsequently reused when applying the prioritisation criteria associated with the scoring systems under analysis. By relying on the same vulnerability list throughout all stages, the methodology ensures that the observed differences in prioritisation outcomes are attributable solely to the applied criteria and not to variations in the underlying data.

4.1.2 Input 2: Define the ideal prioritisation criteria

A set of reference prioritisation criteria is defined by the user to represent a baseline or ideal ordering of vulnerabilities. These criteria express a prioritisation perspective against which other approaches can be evaluated, without imposing any constraints on how the criteria are formulated or combined. For example, the reference prioritisation may be based on the likelihood of exploitation, potential impact severity, asset criticality, exploit availability, remediation urgency, or a combination of these factors.

Importantly, the choice of reference criteria is left to the evaluator and can be adapted according to the specific aspect of prioritisation being assessed, such as risk exposure, operational impact, or timeliness of remediation. The reference criteria are used solely to generate the baseline ordering, which serves as the point of comparison for all other prioritisation results produced by the methodology.

4.1.3 Input 3: Define Scoring Systems prioritisation

Each prioritisation approach under evaluation is associated with its own set of criteria, defined by the user. These criteria may differ across approaches and are not restricted in terms of structure, scale, or underlying rationale.

Applying these criteria to the vulnerability list yields one prioritised list per approach, enabling a comparative assessment relative to the reference ordering.

4.1.4 Step 1: Order the ideal list according to chosen criteria

A reference ordering of the vulnerability list is produced by applying the reference prioritisation criteria (Section 4.1.2). This ordering represents the baseline against which other prioritisation results are assessed. No assumptions are made regarding the method used to derive the ordering, as long as a total ranking of the vulnerabilities can be obtained.

4.1.5 Step 2: Order Scoring Systems prioritisation list according to chosen criteria

Using the same vulnerability list, additional orderings are generated by applying the prioritisation criteria (Section 4.1.3) associated with each evaluated approach. Each resulting ordering reflects the prioritisation logic defined by the corresponding criteria. These orderings constitute the primary artefacts to be compared with the reference ordering in subsequent steps.

4.1.6 Step 3: Compute distance between the Scoring System list and the ideal list

The comparison between different prioritisations is based on the positional differences between a scoring system priority list and a reference priority list. Each prioritisation is represented as an ordering of the same set of vulnerabilities, allowing a direct position wise comparison, as described in detail in Section 4.3.

To quantify these differences, the Euclidean distance between the two priority lists is computed, as formalised in Equation 4.2. This distance captures the aggregate magnitude of positional deviations across the entire list, assigning greater weight to larger ranking discrepancies. To ensure comparability across different prioritisations, the computed distance is subsequently normalised.

The proposed methodology does not impose constraints on how ties within a scoring system priority list are resolved. Any tie-breaking strategy that yields a valid prioritisation may be adopted by the user.

4.1.7 Step 4: Compute $\mathbb{Q}$ Index

Building upon the normalised distance values obtained in the previous step, a comparative index, denoted as $\mathbb{Q}$, is computed to enable direct comparison between scoring systems. The $\mathbb{Q}$ index is defined as a relative measure that expresses the improvement of one scoring system over another with respect to the reference prioritisation, as detailed in Section 4.4.

By relying on ratios of normalised distances, the $\mathbb{Q}$ index remains scale free and symmetric, allowing prioritisation approaches to be compared independently of the absolute magnitude of their distance values.

4.1.8 Step 5: Compare scoring systems

Using the $\mathbb{Q}$ index values, the scoring systems under evaluation are compared in terms of their relative alignment with the reference prioritisation. Positive, negative, or zero values of Q indicate superior, inferior, or equivalent prioritisation performance, respectively.

4.1.9 Output 1: Visualisation of comparative Index Q statistics

The output of the methodology consists of comparative statistics derived from the computed $\mathbb{Q}$ index values. These statistics provide a concise and interpretable summary of the relative performance of the evaluated scoring systems.

The results may be presented in visual form and constitute the basis for analysing and discussing differences in prioritisation quality across scoring systems. The generated graphics provide an intuitive view of the relative performance of the evaluated scoring systems, facilitating the identification of trends, differences, and comparative advantages.

4.2 Definition of the Ideal Prioritisation

To establish the reference prioritisation (Input 2, Section 4.1.2), assumed to be ideal, it is first necessary to reflect on the objective pursued when priorities are defined. Depending on the context, an ideal ordering could be established in several ways: by the potential impact of each vulnerability on the organisation's assets, by the cost or feasibility of remediation, or by the likelihood that a vulnerability will be exploited in the wild. In this case study, the performance of SecScore, EPSS, and CVSS will be compared. Since SecScore aims to adjust the vulnerability score based on the likelihood of an exploit emerging for it, the ideal prioritisation was defined according to the date on which an exploit first appeared for each vulnerability. In other words, the vulnerability whose exploit emerged earlier should be prioritised. However, this prioritisation criterion cannot be applied indiscriminately. Certain vulnerabilities, due to other characteristics, cannot be treated at the same level for the purposes of this criterion. For example, it would not be reasonable to assign a higher priority to a vulnerability V_A with a CVSS score of 1 over a vulnerability V_B with a CVSS score of 9 solely because the probability of exploits emerging for V_A is higher.

One way to better address this issue in defining the reference prioritisation is to first prioritise vulnerabilities by severity strata — adopting, for example, the qualitative severity ranges defined by CVSS [27] — and then prioritising them based on the probability of an exploit emerging. Accordingly, in the prioritisation adopted in this study as a reference, the set of vulnerabilities is initially categorised into four levels: Critical, High, Medium, and Low, based on the CVSS base score assigned to each CVE. This initial categorisation follows the CVSS qualitative severity classification [27], as presented in Table 2.11. After this partition, the final reference order is obtained by concatenating the four strata in the precedence order $Critical \succ High \succ Medium \succ Low$.

Subsequently, the Reference Priority List ($\mathbb{R}$) is constructed (Step 1, Section 4.1.4), which in this study represents the ideal order of prioritisation and serves as the reference (ground truth) for evaluating the performance of SecScore and other scoring systems. In this list, the CVEs already grouped by the strata in Table 2.11 are ordered within each stratum chronologically by Exploit-Date, from the earliest to the most recent. This ordering reflects the ideal scenario in which vulnerabilities are prioritised according to the moment when exploit code becomes available, assigning higher priority to those exploited earlier. To guarantee that this ordering is unique, each vulnerability V is associated with an ordering key, as shown in Equation 4.1.

$$\boldsymbol{\kappa}(V) \;=\; \big(\,\tau_{\text{exploit}}(V)\,;\; \tau_{\text{cve}}(V),\; \text{CVE}(V)\,\big) \tag{4.1}$$

Where:

- $\boldsymbol{\kappa}(V)$ is the final reference order,
- $\tau_{\text{exploit}}(V)$ is the earliest observed exploit date,
- $\tau_{\text{cve}}(V)$ is the CVE publication date used as the first tie-breaker,
- $\text{CVE}(V)$ is the CVE identifier used as the final tie-breaker.

Vulnerabilities for which no exploit date is known are excluded from the exploit-based reference, ensuring that all items included in $\mathbb{R}$ have a complete comparison key.

Thus, a list that arranges vulnerabilities by their actual exploitation dates provides a natural and meaningful reference point for assessing the extent to which SecScore can anticipate real-world exploitation events. By comparing the prioritisation generated by SecScore with a list that mirrors the chronological order of exploitations, it is possible to measure how early the system assigns priority to vulnerabilities that are ultimately exploited — an essential indicator for proactive risk mitigation.

It is important to emphasise that, although the case study adopted in this work to evaluate the proposed method focuses on the temporal aspect of exploit development dynamics, the method can also be applied to analyses in which prioritisation is guided by other aspects of vulnerabilities. In such cases, these alternative perspectives should be reflected in the construction of the reference list $\mathbb{R}$, where only the first component of the ordering key changes, while the deterministic tie-breaking structure remains identical.

4.3 Measuring the Distance between Different Prioritisations

The next sets of lists to be evaluated are the Priority Lists of the Scoring Systems ($\mathbb{P}_{\textit{scoring system}}$), in which the CVEs are ordered from highest to lowest according to the scores assigned by each metric (SecScore, EPSS, or CVSS). This ordering step (Step 2, Section 4.1.5) defines the prioritisation produced by each scoring system and provides the basis for subsequent comparison with the reference list. When a scoring system assigns the same score to two or more CVEs, a tie-breaking criterion is required, as identical scores allow for multiple valid priority positions. To address this ambiguity, multiple lists can be generated by randomising the order of CVEs with identical scores, thereby creating several plausible prioritisation scenarios for each scoring system. This process enables a more robust and realistic evaluation of each system's performance, taking into account the uncertainty introduced by the occurrence of identical scores across different vulnerabilities.

To enable a fair comparison of the prioritisation resulting from different scoring systems, the positional differences between the Reference Priority List and each Scoring System Priority List are quantified by calculating the Euclidean distance between them (Step 3, Section 4.1.6), as illustrated in Equation 4.2:

$$\boldsymbol{\varepsilon}(\mathbb{P}_{\textit{scoring system}}, \mathbb{R}) = \sqrt{\sum_{i=1}^{n} (p_i - r_i)^2} \tag{4.2}$$

Where:

- ε denotes the total Euclidean distance,
- $\mathbb{P}_{\textit{scoring system}} = \{p_i \mid i \in \{1, 2, \ldots, n\}\}$ where p_i is the position of CVE i in a given scoring system,
- $\mathbb{R} = \{r_i \mid i \in \{1, 2, \ldots, n\}\}$ where r_i is the position of CVE i in the Reference List $\mathbb{R}$,
- n is the total number of CVEs considered.

This metric penalises large positional deviations more heavily, which aligns with the evaluation goal of detecting substantial ranking errors. To ensure that the distance metric is comparable across different lists and scenarios (as explained later in Section 4.4), the calculated Euclidean distance ε is scaled by a reference limiting factor that depends solely on the number of CVEs n.

The raw Euclidean distance is divided by a theoretical limiting factor on the distance, which is a function of n only. Since each CVE position can differ by at most $(n-1)$ positions between two lists, the maximum possible squared shift for any CVE is $(n-1)^2$. Assuming all n CVE simultaneously attain this maximum deviation, it yields the following limiting factor on Equation 4.3, noting that the resulting value never exceeds 1:

$$\varepsilon_{\max}(n) = \sqrt{n(n-1)^2} = (n-1)\sqrt{n}. \tag{4.3}$$

The scaled Euclidean distance between a Scoring System Priority List and the Reference Priority List is then given by Equation 4.4:

$$\varepsilon_{\text{norm}} = \frac{\varepsilon(\mathbb{P}_{\textit{scoring system}}, \mathbb{R})}{\varepsilon_{\max}(n)} = \frac{\sqrt{\sum_{i=1}^{n}(p_i - r_i)^2}}{\sqrt{n(n-1)^2}} \tag{4.4}$$

In cases where multiple Scoring System Priority Lists are generated due to score ties, the Euclidean distance is first computed for each generated list and then averaged. Let $\bar{\varepsilon}$ denote this average, the final scaled distance is then obtained as illustrated in Equation 4.5:

$$\varepsilon_{\text{norm}} = \frac{\bar{\varepsilon}}{\varepsilon_{\max}(n)}. \tag{4.5}$$

This scaling yields values in the interval $[0, 1]$, where 0 indicates perfect agreement with the Reference Priority List and values closer to 1 indicate increasingly large aggregate positional deviations. Accordingly, all scoring systems must operate over the same set of CVEs to guarantee that the resulting distance values remain meaningful and comparable.

4.4 Comparison between Vulnerability Scoring Systems

Finally, to comparatively assess the quality of the prioritisation produced by different scoring systems (Step 4 and Step 5, Sections 4.1.7 and 4.1.8), Equation 4.6 is used:

$$\mathbb{Q} = \left(1 - \frac{\varepsilon_{\text{norm}}(\mathbb{P}_{\text{scoring system 1}}, \mathbb{R})}{\varepsilon_{\text{norm}}(\mathbb{P}_{\textit{scoring system 2}}, \mathbb{R})}\right) \times 100\% \tag{4.6}$$

Where:

- $\hat{\varepsilon}(\mathbb{P}_{\text{scoring system 1}}, \mathbb{R})$ denotes the normalised Euclidean distance between scoring system 1 (e.g., SecScore) and the Reference List $\mathbb{R}$,
- $\hat{\varepsilon}(\mathbb{P}_{\text{scoring system 2}}, \mathbb{R})$ denotes the normalised Euclidean distance between scoring system 2 (e.g., CVSS) and the Reference List $\mathbb{R}$.

A $\mathbb{Q}$ value greater than zero indicates that $\mathbb{P}_{\text{scoring system 1}}$ prioritises better than $\mathbb{P}_{\text{scoring system 2}}$. If the $\mathbb{Q}$ value is negative, then $\mathbb{P}_{\text{scoring system 2}}$ achieves better prioritisation. A $\mathbb{Q}$ value equal to zero means there is no improvement, and both systems' prioritisation is aligned with the Reference List $\mathbb{R}$. Because the index is symmetric (swapping systems 1 and 2 only changes the sign of the index) and scale-free (since it compares the ordering of vulnerabilities independently of the scale of the metrics used by different systems), its interpretation does not depend on the magnitude of the distances involved, only on their relative size. When the denominator in Equation 4.6 is zero, the corresponding $\mathbb{Q}$ value is set to zero by convention. In addition, the computed $\mathbb{Q}$ values are used to generate graphical representations (Output 1, Section 4.1.9) that support the visual analysis of the comparative results.

Through these metrics, it is possible to compare prioritisation results in a quantitative and systematic manner. This approach provides empirical evidence to determine whether the prioritisation offered by a scoring system such as SecScore delivers a consistent advantage over other scoring systems.

Chapter 5

Experimental Evaluation

This chapter presents the empirical evaluation of different vulnerability scoring systems used for VMP. Building on the methodology defined in Chapter 4, we apply the evaluation framework to real-world vulnerability data in order to compare the performance of CVSS, SecScore, and EPSS. The chapter is organised into four parts. First, we describe the dataset and experimental setup, clarifying the scope and temporal coverage of the analysis. Second, we conduct a case study on vulnerabilities disclosed in May 2005 to illustrate the behaviour of the scoring systems in a controlled context. Third, we extend the evaluation to a longitudinal study across multiple time windows to assess consistency and robustness. Finally, we show the potential of the method to assess the sensitivity of SecScore to its parameterisation, highlighting its practical implications for vulnerability management.

5.1 Dataset

The dataset used to evaluate SecScore is a validated and extended version of the dataset originally presented by the authors in their SecScore article [35]. Initially, the dataset comprised information on approximately 27,000 CVEs. For each entry, the collected data included the CVE assignment date (ranging from 2001 to 2022), the existence of a publicly known exploit, the assigned CVSS base score, the SecScore score, the time difference (in weeks and days) between the CVE assignment date and the exploit disclosure date, the type of vulnerability, the detection tool used, and the affected platform.

From this dataset, all entries with a CVSS score of -1 and all duplicate CVEs have been removed. These duplicate CVEs appeared in the dataset due to updates in the Exploit-Date column, where a more recent, publicly known exploit might have been published for the same CVE. To address this, only the earliest instance of each CVE was retained, and all updated entries were excluded.

To enable the analysis of the impact of different time windows, the dataset was also extended to include the SecScore score for each time window for every CVE. These time windows, which represent the periods of data considered for the SecScore calculation, range from 6 months to 20 years. For example, a 6 month time window means that only data from the most recent 6 months

(e.g., from July 2022 to January 2022) was used to calculate the corresponding SecScore. The same logic applies to the other time windows.

Subsequently, the EPSS score was also added for each CVE to enable a comparison between EPSS and SecScore, along with the Exploitability Score. However, the latter was ultimately excluded from the analysis presented in the following sections. The Exploitability Score was initially considered for ordering the Reference List $\mathbb{R}$ or as a secondary ordering criterion in addition to the Exploit-Date, but it was found to have little to no impact in this context. Nevertheless, the variable was retained in the dataset, as it may prove useful for future iterations. Therefore, the resulting dataset comprises approximately 22,000 vulnerabilities, incorporating the extensions and modifications described above.

5.2 Analysis of the Week With Most Cases

The following tables (5.1, 5.4, 5.5, 5.7) present the results obtained from the comparison of the SecScore and CVSS prioritisation lists with the Reference Priority List for the week with the highest number of cases. For CVSS, ten prioritisation lists (CVSS 1 to CVSS 10) are considered to account for the presence of identical scores assigned to multiple CVEs. Because equal scores permit multiple valid priority positions, the ordering of vulnerabilities with identical CVSS values was randomised, generating several plausible prioritisation scenarios. This approach avoids reliance on a single arbitrary tie-breaking rule and provides a more robust assessment of CVSS prioritisation behaviour. This week was initially selected to capture a realistic yet challenging prioritisation scenario. It corresponds to the first week of May 2005 and comprises 220 CVEs, of which 9 were categorised as Critical, 77 as High, 123 as Medium, and 11 as Low.

The evaluation results demonstrate that SecScore outperforms CVSS in prioritising, especially for the Critical and High criticality levels of vulnerabilities. Its ability to maintain proximity to both optimal lists, as reflected in consistently lower Euclidean Distances, demonstrates its effectiveness in identifying vulnerabilities that need immediate attention.

	Euclidean Distance Normalised	
	Optimal List	Deterministic Optimal List
SecScore	0.497	0.493
CVSS (Average)	0.529	0.559

Table 5.1: Comparison of SecScore and CVSS Lists with the Optimal scenarios for Critical Qualitative Rating

Tables 5.1 and 5.2 show that SecScore consistently records a lower Euclidean Distance and the smallest Normalised Euclidean Distance when compared with CVSS. This closer alignment with the optimal prioritisation indicates that SecScore's dynamic approach is more effective for critical qualitative ratings.

Specifically, SecScore outperforms CVSS by approximately 6.1% for the optimal list and 11.9% for the deterministic optimal list. The fact that the Average CVSS Euclidean Distance is

	Euclidean Distance		Euclidean Distance Normalized	
	Optimal List	Deterministic Scenario	Optimal List	Deterministic Scenario
SecScore	8.94427191	11.83215957	0.496903995	0.493006649
CVSS	12.32882801	12.56980509	0.587087048	0.598562147
CVSS 1	10.19803903	9.273618495	0.566557724	0.515201028
CVSS 2	8.124038405	7.348469228	0.451335467	0.489897949
CVSS 3	9.38083152	7.071067812	0.446706263	0.589255651
CVSS 4	7.615773106	9.38083152	0.634647759	0.521157307
CVSS 5	9.899494937	9.695359715	0.549971941	0.461683796
CVSS 6	9.899494937	9.16515139	0.471404521	0.763762616
CVSS 7	8.485281374	9.16515139	0.471404521	0.509175077
CVSS 8	11.13552873	10.95445115	0.618640485	0.521640531
CVSS 9	9.486832981	7.874007874	0.451753951	0.656167323
CVSS 10	10.29563014	9.486832981	0.571979452	0.527046277

Table 5.2: Results when comparing SecScore and CVSS Lists with the Optimal scenarios for Critical Qualitative Rating

higher than SecScore's distance to both lists underscores the advantage of SecScore's nuanced, adaptive prioritisation approach over the static scoring system of CVSS.

In Tables 5.3 and 5.4, SecScore again attains lower Euclidean Distances — both raw and normalised — than every CVSS variant. In both instances, SecScore's Normalised Euclidean Distance is smaller than the Average CVSS Euclidean Distance, showing a prioritisation that more closely matches the optimal lists.

	Euclidean Distance		Euclidean Distance Normalized	
	Optimal List	Deterministic Scenario	Optimal List	Deterministic Scenario
SecScore	223.7319825	223.1232843	0.359107404	0.368510987
CVSS	358.5219659	334.2962758	0.536097491	0.536571779
CVSS 1	286.6600774	259.2489151	0.460111939	0.447638724
CVSS 2	266.2442488	252.7607564	0.446196295	0.423599434
CVSS 3	259.0791385	216.6748716	0.421782971	0.446880595
CVSS 4	281.7729582	256.7177438	0.442820661	0.457120345
CVSS 5	278.2193379	266.773312	0.452943373	0.453755827
CVSS 6	251.7419314	251.4955268	0.402460494	0.403670073
CVSS 7	283.3901904	253.5586717	0.459733555	0.451495194
CVSS 8	249.4313533	233.1608887	0.398751834	0.415174207
CVSS 9	292.8856432	293.867317	0.489168473	0.492489542
CVSS 10	264.1439002	240.3455845	0.401359882	0.38577354

Table 5.3: Results when comparing SecScore and CVSS Lists with the Optimal scenarios for High Qualitative Rating

The measured improvements are about 19.6% for the optimal list and 17.5% for the deterministic optimal list. Together, these results indicate that SecScore more effectively identifies the high-severity vulnerabilities that require prompt attention.

	Euclidean Distance Normalised	
	Optimal List	Deterministic Optimal List
SecScore	0.359	0.369
CVSS (Average)	0.446	0.447

Table 5.4: Comparison of SecScore and CVSS Lists with the Optimal scenarios for High Qualitative Rating

For the Medium Level, SecScore's normalised Euclidean distance is higher than distance for CVSS, as shown in Tables 5.5 and 5.6.

	Euclidean Distance Normalised	
	Optimal List	Deterministic Optimal List
SecScore	0.465	0.436
CVSS (Average)	0.448	0.354

Table 5.5: Comparison of SecScore and CVSS Lists with the Optimal scenarios for Medium Qualitative Rating

In this level, SecScore's performance does not consistently outperform CVSS. Specifically, the Euclidean Distance for SecScore is 3.9% worse than CVSS for the optimal list and 23.1% worse for the deterministic optimal list. The larger deviation suggests that SecScore may be over-influencing the prioritisation of medium priority vulnerabilities, causing a misalignment with the optimal prioritisation.

	Euclidean Distance		Euclidean Distance Normalized	
	Optimal List	Deterministic Scenario	Optimal List	Deterministic Scenario
SecScore	552.946652	231.982758	0.465115454	0.435774601
CVSS	487.8831827	29.69848481	0.418961382	0.243438381
CVSS 1	559.1457771	208.2930628	0.446163512	0.37562306
CVSS 2	551.6103697	198.8114685	0.452154837	0.365841355
CVSS 3	572.6045756	211.3338591	0.468544081	0.373633968
CVSS 4	561.8576332	201.2063617	0.432999972	0.329857576
CVSS 5	563.5902767	213.0070421	0.461974763	0.349203603
CVSS 6	561.6368934	210.8316864	0.452152618	0.387960265
CVSS 7	552.8887772	204.8804529	0.440372878	0.384863507
CVSS 8	551.4653208	194.2987391	0.44396387	0.389318385
CVSS 9	560.6817279	209.0502332	0.462979834	0.355649523
CVSS 10	561.5442992	205.3143931	0.44414688	0.33659228

Table 5.6: Results when comparing SecScore and CVSS Lists with the Optimal scenarios for Medium Qualitative Rating

Finally, for the Low Level, SecScore once again achieves lower distances. In Tables 5.7 and 5.8, we can see that SecScore outperforms CVSS, but the improvement is not substantial for the optimal list since the difference is marginal. SecScore improves by approximately 1.5% for the optimal list and 18.5% for the deterministic optimal list. The small difference in the optimal list comparison suggests that for low severity vulnerabilities, both SecScore and CVSS provide relatively close prioritisations. However, the significant improvement in the deterministic optimal list comparison highlights that SecScore is better at distinguishing low priority vulnerabilities when fewer uncertainties are involved.

	Euclidean Distance Normalised	
	Optimal List	Deterministic Optimal List
SecScore	0.513	0.391
CVSS (Average)	0.521	0.488

Table 5.7: Comparison of SecScore and CVSS Lists with the Optimal scenarios for Low Qualitative Rating

	Euclidean Distance		Euclidean Distance Normalized	
	Optimal Scenario	Deterministic Scenario	Optimal Scenario	Deterministic Scenario
SecScore	11.91637529	6.480740698	0.513274619	0.390803368
CVSS	9.591663047	2.449489743	0.431747313	0.369274473
CVSS 1	14.89966443	8.602325267	0.499157539	0.648424664
CVSS 2	12.80624847	7.483314774	0.643538199	0.564076075
CVSS 3	11.66190379	4.472135955	0.439524536	0.449466575
CVSS 4	13.6381817	6.164414003	0.476319395	0.371728151
CVSS 5	13.11487705	7.874007874	0.659047369	0.47482054
CVSS 6	14	6.92820323	0.435516387	0.522232968
CVSS 7	12.40967365	4.69041576	0.430018255	0.471404521
CVSS 8	14	7.745966692	0.55994964	0.467099366
CVSS 9	12.16552506	6.92820323	0.561088746	0.522232968
CVSS 10	14.83239697	8.485281374	0.595803516	0.511681719

Table 5.8: Results when comparing SecScore and CVSS Lists with the Optimal scenarios for Low Qualitative Rating

5.3 Comparison Between SecScore and CVSS Considering the Entire Dataset

In this subsequent case study, the prioritisation provided by SecScore is compared with that provided by CVSS. Each priority list used in the evaluation consists of the vulnerabilities that appeared over the course of a given week. The evaluation process is then applied on a weekly basis across the entire dataset.

Figure 5.1 presents a violin plot showing the distribution of the percentage improvements

(calculated according to Equation 4.6) achieved by SecScore over CVSS, across the four severity levels defined in Table 2.11. These improvements reflect the enhanced accuracy with which SecScore prioritises vulnerabilities compared with CVSS, using the reference list R as a baseline.

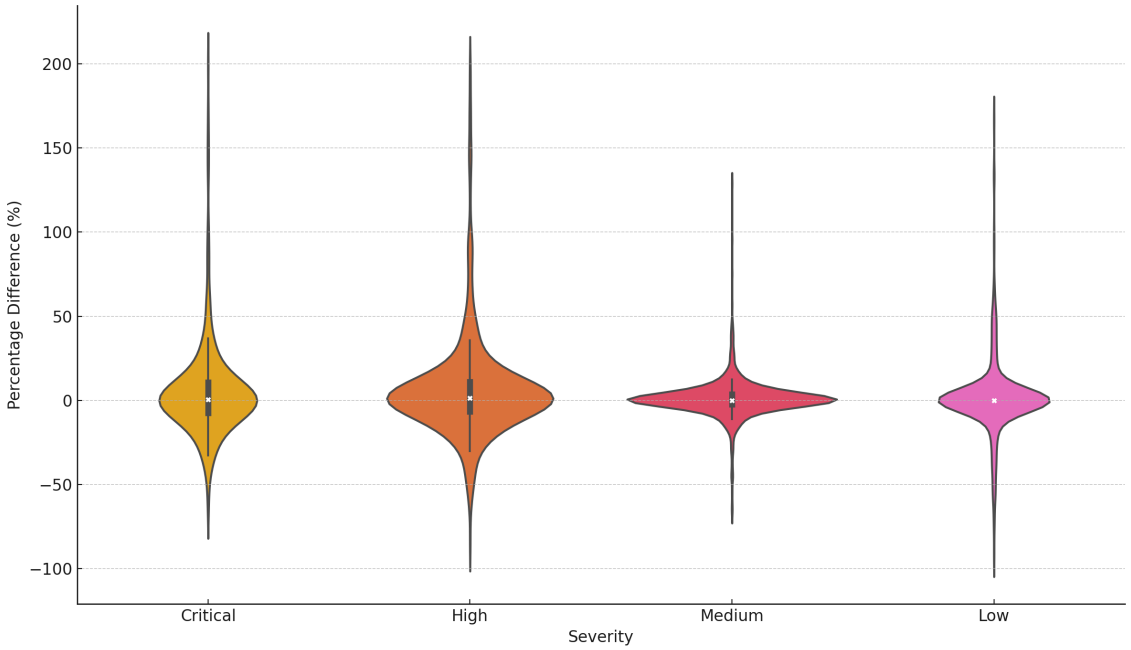


Figure 5.1: Percentage improvement of SecScore over CVSS by severity level

The data show that SecScore provides more significant improvements in the Critical and High severity categories. Although the median improvements are close to zero, the distributions for both categories display pronounced positive skewness and wide variance, with some individual cases recording improvements greater than 200%. This indicates that SecScore is often able to prioritise high-risk vulnerabilities much earlier than CVSS.

These results are further supported by the average improvement values summarised in Table 5.9.

SecScore achieves an average improvement of 6.30% for High severity vulnerabilities and 5.24% for Critical vulnerabilities. In contrast, the Medium and Low severity categories show average improvements of only 0.95% and 1.49%, respectively.

Severity	Average Improvement (%)
Critical	5.24
High	6.30
Medium	0.95
Low	1.49

Table 5.9: Average Improvement by Severity

The narrower distributions and lower average values in these lower-severity categories suggest that SecScore's prioritisation is closely aligned with that of CVSS for less critical vulnerabilities.

This indicates that SecScore concentrates its differentiation where it is most relevant — identifying and elevating with greater precision those vulnerabilities with a higher likelihood of active exploitation.

Overall, these results demonstrate that SecScore consistently improves prioritisation accuracy in high-severity contexts, reinforcing its value as a more effective and risk-sensitive scoring system for vulnerability management.

5.4 Evaluation of the Time Window Used for Defining SecScore Parameters

In this third case study applying the proposed method, the time window used to define the statistical parameters of SecScore is evaluated. In other words, the analysis assesses how the time window employed in calculating the parameters of SecScore's asymmetric Laplace distribution impacts its performance in vulnerability prioritisation.

To investigate how the temporal scope of historical data influences SecScore's ability to prioritise vulnerabilities effectively, twenty-two versions of SecScore were computed using different time windows: 6 months and then from 1 year to 21 years (the year 21 is referred to in the following tables and graphs as "all" since it encompasses every year in the dataset). Note that the window size defines the subset of vulnerabilities used to estimate the parameters of the asymmetric Laplace distribution — which, in SecScore, models the probability of an exploit code emerging. For each window size, it is measured how closely the prioritisation provided by SecScore's statistical model aligns with the ideal ordering R, using $\varepsilon_{\text{norm}}(P_{\text{SecScore}}, R)$ as the evaluation metric.

Tables 5.10, 5.11, and 5.12 present the average improvement, standard deviation, and skewness values by severity level and time span, respectively. In all three tables, values highlighted in blue correspond to the lowest value observed within each stratum, while those highlighted in yellow correspond to the highest. Table 5.13 reports the same statistical measures for EPSS, organised by severity level. All of these values are also visualised in Figure 5.2 for easier interpretation.

The average captures the central point of improvement of SecScore in prioritising vulnerabilities. For Critical severity, the average value increases substantially across time windows, with the lowest improvement of 0.49% at 2 years and a maximum of 7.55% at 13 years. However, this upward trend does not persist indefinitely. After peaking at 13 years, the average drops to 5.28% at 14 years, dips again, and then rises once more before dropping at the end. This non-monotonic pattern indicates that while long-term historical windows can yield substantial improvements over CVSS, the enhancement is not strictly increasing beyond a certain horizon. It suggests that overly long exposure windows may introduce noise or dilute the relevance of exploit signals, calling for a balanced selection of temporal scope in model calibration.

For High severity, the mean initially dips (e.g., 0.66% at 1 year—the lowest), before rising sharply to a peak of 9.21% at 14 years. This upward trend suggests high sensitivity to both short- and long-term shifts in exploit dynamics. Notably, the increase between years 3 and 5 is particularly steep, with values rising from 1.35% to over 5.16%, indicating a concentrated period of

	Severity			
Time Span	Critical	High	Medium	Low
6m	1.048%	1.094%	0.660%	-1.895%
1y	1.424%	0.660%	0.744%	0.293%
2y	0.490%	0.756%	0.103%	1.737%
3y	0.538%	1.344%	-0.096%	1.224%
4y	4.077%	4.726%	2.667%	9.721%
5y	5.897%	5.159%	0.542%	12.934%
6y	6.048%	6.939%	0.127%	9.269%
7y	4.093%	8.472%	0.940%	11.717%
8y	6.625%	8.639%	1.071%	3.102%
9y	2.957%	6.604%	1.369%	4.578%
10y	4.744%	5.807%	1.467%	4.053%
11y	7.028%	6.141%	1.830%	4.125%
12y	6.360%	6.865%	1.522%	5.904%
13y	7.548%	8.438%	1.043%	3.090%
14y	5.282%	9.210%	1.818%	7.080%
15y	4.220%	7.291%	1.391%	2.932%
16y	4.782%	7.434%	1.857%	6.044%
17y	3.651%	6.023%	1.623%	4.319%
18y	4.967%	6.510%	1.986%	7.011%
19y	6.046%	7.829%	2.182%	10.532%
20y	6.698%	7.077%	1.439%	4.657%
all	4.994%	7.121%	1.307%	9.234%

Table 5.10: Comparison of Average Improvement for Each Time Window by Stratum, where blue correspond to the lowest value and yellow correspond to the highest

performance gain. However, beyond the 14-year peak, the average begins to decline slightly, indicating that excessively long historical windows may not necessarily translate to proportional gains and could potentially introduce noise. This reinforces the idea that while longer windows can provide improved prioritisation, there may be diminishing returns or even minor reversals beyond a certain threshold.

In the case of Medium severity, the average exhibits a sharp local fluctuation between 3 and 4 years, moving from a minimum value of -0.10% to a maximum of 2.67%. However, beyond this abrupt jump, the values tend to stabilise within a narrower range between approximately 0.5% and 2% across most remaining time windows. This suggests that while medium-severity vulnerabilities are sensitive to short-term shifts in sampling, their long-term behaviour remains relatively moderate compared to other severity levels.

Most notably, Low severity experiences the most striking increase in average improvement, from a minimum of -1.90% at 6 months to a maximum of 12.93% at 5 years — the highest mean improvement of all severities. This steep upward trend suggests that low-severity vulnerabilities are heavily underprioritised in short-term assessments and that historical perspective significantly enhances their recognition. Notably, the rapid increase occurs within the first five years, after

which the average fluctuates widely, with several windows showing values well above 5% — including peaks at 7%, 9%, and over 11% — while others drop below 4.5%. This pattern suggests that a substantial portion of the performance gain for low-severity vulnerabilities is realised within the first five years of historical expansion. Beyond this point, the average values no longer follow a simple trend; instead, they fluctuate markedly, with some windows exhibiting continued strong improvements — up to 11.72% at 7 years. This volatility highlights a dynamic interplay between signal accumulation and noise introduction as historical breadth increases. It also underscores the risk of relying on short windows, which systematically undervalue low - severity vulnerabilities. Therefore, careful tuning of temporal scope is essential to capture hidden, but persistent exploitability signals without overfitting on episodic outliers.

	Severity			
Time Span	Critical	High	Medium	Low
6m	16.014	25.055	12.722	20.453
1y	18.045	17.147	12.843	18.266
2y	14.427	13.180	11.104	21.850
3y	25.161	16.888	7.015	17.983
4y	27.085	28.390	27.895	42.863
5y	29.102	31.923	15.260	61.711
6y	31.266	63.223	11.955	46.637
7y	29.364	82.061	12.983	55.309
8y	34.845	66.941	13.697	44.593
9y	25.014	43.233	14.167	37.777
10y	26.158	33.364	14.590	35.775
11y	37.702	38.192	16.373	42.562
12y	35.723	35.571	13.906	40.737
13y	32.593	47.277	12.905	35.938
14y	31.474	49.643	18.015	52.979
15y	26.852	40.235	13.501	30.533
16y	30.024	37.227	16.161	38.248
17y	25.094	32.513	15.429	35.312
18y	28.890	34.190	16.244	45.962
19y	40.402	62.870	16.421	63.867
20y	42.698	43.065	16.398	46.630
all	33.738	41.249	13.436	43.476

Table 5.11: Comparison of Standard Deviation for Each Time Window by Stratum, where blue correspond to the lowest value and yellow correspond to the highest

Standard deviation measures the spread in SecScore's improvement. Higher values reflect more variability across the vulnerability set. In Critical severity the variability increases with time, ranging from a minimum of 14.43 (2 years) to a maximum of 42.70 (20 years). This increasing dispersion reflects a widening spread in prioritisation scores as more historical data is incorporated, suggesting that vulnerabilities assessed over longer horizons exhibit greater heterogeneity in exploit likelihood. The progressive rise in standard deviation indicates that, although average

prioritisation improves, the uncertainty in ranking grows as well. Notably, the growth in variability appears smoother than in other severity levels, highlighting the more stable accumulation of long-term exploit signals in critical vulnerabilities.

In the case of High severity, it is demonstrated the most dramatic range in dispersion — from 13.18 (minimum at 2 years) to 82.06 (maximum at 7 years). This spike reflects a temporal cluster of highly varied vulnerability outcomes within the high-severity category. After this spike, dispersion remains elevated but becomes more stable, implying that while high-severity vulnerabilities retain some variability, the most extreme heterogeneity is concentrated in this specific historical slice.

For Medium severity, the standard deviation fluctuates moderately. This abrupt increase of nearly 300% in just one year (between 7.01 (3 years) and 27.90 (4 years)) is indicative of substantial sensitivity to temporal boundaries in the data. However, it is important to note that following this jump, the standard deviation remains relatively contained, fluctuating between 10 and 20 for most time windows. This suggests that while medium-severity vulnerabilities are prone to sharp reactions when historical coverage shifts abruptly, their variability stabilises somewhat in the long term. This pattern implies that initial volatility may stem from transient factors, while the underlying risk landscape becomes more predictable as more data accumulates. The sharp volatility in dispersion suggests that medium-severity vulnerabilities do not exhibit stable exploit patterns across time, but instead respond acutely to the specific subset of vulnerabilities captured within a narrow historical window.

Low severity starts at a moderate 20.45 (6 months) and climbs to 63.87 at 19 years. The high variance aligns with the wide range of prioritisation improvements observed, and reflects the increased heterogeneity of this severity class over long periods.

Skewness measures the asymmetry of the distribution of improvement values. Positive skew means most vulnerabilities exhibit modest improvements, while a few show very large gains. For Critical severity, skewness reaches a pronounced peak of 11.84 at 3 years, then drops to 1.93 at 10 years, and stabilises between 2.8 and 5 thereafter. The sharp early asymmetry implies concentrated benefits for few vulnerabilities, while longer windows lead to more balanced performance improvements.

High severity skewness fluctuates substantially over time, peaking at 14.59 at 19 years and an early high of 14.50 at 6 months. However, this trend is far from linear. The skewness drops to a relative low of 2.04 at 2 years and shows considerable variation across time windows, with other notable dips at 4 and 5 years.

In the case of Medium severity, it has the highest skew overall — 17.39 at 4 years, with a low of -0.07 at 13 years — indicating that performance benefits are unevenly distributed and highly dependent on the time window.

For Low Severity, it rises from 0.22 at 6 months to 5.93 at 20 years. The trend indicates that improvements become more asymmetric as more data is added, likely due to outlier vulnerabilities driving stronger gains.

	Severity			
Time Span	Critical	High	Medium	Low
6m	3.697	14.504	11.374	0.220
1y	3.154	9.660	7.421	2.079
2y	3.463	2.038	9.387	0.716
3y	11.843	9.861	0.946	1.207
4y	3.480	2.682	17.386	2.699
5y	3.186	4.044	6.333	4.961
6y	3.225	12.335	1.238	2.667
7y	3.310	12.963	4.144	3.210
8y	4.902	13.476	3.654	4.040
9y	2.853	6.411	3.135	2.041
10y	1.931	4.575	3.166	1.568
11y	4.429	6.293	5.724	3.131
12y	5.918	4.791	3.013	1.957
13y	3.912	6.285	-0.074	1.394
14y	2.833	7.891	6.420	4.149
15y	2.379	7.182	0.249	0.439
16y	4.095	4.760	3.268	2.854
17y	2.602	4.574	2.330	1.536
18y	3.392	4.828	6.962	3.129
19y	6.426	14.589	5.155	4.462
20y	8.313	6.103	8.817	5.926
all	5.726	7.363	2.672	1.982

Table 5.12: Comparison of Skewness for Each Time Window by Stratum, where blue correspond to the lowest value and yellow correspond to the highest

5.5 Comparison between SecScore and EPSS

Regarding EPSS, average improvements differ markedly from those of SecScore across all severity levels. For low severity, EPSS achieves an average of 14.18%, which is higher than any of SecScore's time-window averages. This indicates that EPSS delivers greater improvement in prioritisation accuracy for vulnerabilities classified as low severity. At the medium level, the mean improvement is 0.18%, broadly consistent with SecScore's overall behaviour, though generally at the lower end of its improvement spectrum. In contrast, high severity records a mean of –0.93%, representing a regression in performance relative to CVSS and sharply diverging from SecScore's strong positive improvements. Similarly, the critical category shows an average of –1.49%, again reflecting underperformance compared with CVSS, whereas SecScore consistently achieves gains.

This divergence highlights a fundamental difference in prioritisation philosophy. EPSS appears to emphasise low-severity vulnerabilities, possibly capturing real-world exploitation behaviours that severity-based models such as CVSS tend to overlook. SecScore, by contrast — tuned using the asymmetric Laplace framework — achieves its strongest improvements for high- and critical-severity vulnerabilities, suggesting greater suitability for risk models where impact severity is central. While EPSS surpasses SecScore in the low-severity stratum, SecScore sub-

Severity	Average	Standard Deviation	Skewness
Low	14.182	88.382	3.405
Medium	0.184	38.940	1.909
High	-0.934	38.338	3.325
Critical	-1.493	43.928	1.784

Table 5.13: EPSS Average, Standard Deviation, and Skewness by Severity

stantially outperforms EPSS in all others, particularly in the High and Critical categories. This suggests that SecScore is better aligned with models requiring prioritisation according to potential impact, whereas EPSS provides broader coverage of exploitation likelihood, albeit sometimes at the expense of reduced performance for higher-severity vulnerabilities.

Figure 5.2: Evolution of SecScore Parameters Across Time Windows with EPSS Comparison

EPSS standard deviations are generally higher across all severity levels. In the critical category, the value reaches 43.93, exceeding any of SecScore's results and indicating broader variation in performance improvement. For high severity, the value is 38.34, more stable than SecScore's highly volatile pattern yet still reflecting considerable variability. Medium severity records 38.94, much higher than SecScore's values beyond the 4-year mark, suggesting a broader spread of improvement outcomes. The low category shows the most pronounced dispersion, with a standard deviation of 88.38 — substantially larger than SecScore's already high spread. This indicates that even within low severity, EPSS produces widely varying degrees of improvement, likely due to uneven exploitation signals across individual vulnerabilities. Taken together, these elevated values suggest that EPSS captures a wider spectrum of behaviour, encompassing both improvements and degradations. This may reflect a model architecture that integrates more dynamic, real-world ex-

ploit indicators, leading to greater heterogeneity in outcomes. At the same time, such dispersion may also introduce instability and reduce the consistency of prioritisation.

Across all severity levels, EPSS exhibits lower skewness than SecScore, with values of 1.78 for Critical, 3.32 for High, 1.91 for Medium, and 3.41 for Low. This reflects a more symmetric distribution of relative performance values, suggesting that EPSS spreads its prioritisation improvements more evenly within each severity category and avoids heavy dependence on a few outliers. By contrast, SecScore frequently displays higher skew — particularly at short and long time windows — indicating that improvements often concentrate on a small subset of vulnerabilities. This distinction underscores different modelling philosophies: EPSS favours consistency and balance, while SecScore achieves sharper, targeted gains at the expense of greater asymmetry.

Overall, time spans between 4 and 14 years tend to provide the most effective trade-offs across statistical indicators, whereas selecting windows outside this range introduces either short-term instability or long-term diminishing returns.

Short time windows are characterised by lower average gains, high skewness, and low to moderate volatility. Medium windows, between four and ten years, show marked variability — particularly in the High and Medium severity categories — but generally offer a better balance between average gains and statistical stability. Long windows, ranging from eleven to twenty years, sustain higher average improvements, especially for Low and Critical severities, but they also tend to reintroduce volatility and outlier-driven skewness.

5.6 Final Remarks

The results presented in this chapter demonstrate the effectiveness of the proposed evaluation methodology in analysing and comparing vulnerability scoring systems beyond traditional accuracy based assessments. By framing vulnerability prioritisation as an ordering problem and quantifying deviations from an ideal reference list, the method enables a nuanced examination of how different scoring approaches behave across severity levels and temporal configurations.

The comparative analyses reveal that SecScore consistently improves prioritisation for high and critical severity vulnerabilities, particularly when calibrated with appropriately selected temporal windows, while EPSS exhibits stronger performance in lower-severity categories and greater overall dispersion. These findings illustrate how the methodology not only identifies which scoring systems perform better under specific conditions, but also exposes their underlying prioritisation biases, stability properties, and sensitivity to modelling choices such as historical scope.

More broadly, the proposed method provides a flexible and systematic framework for evaluating vulnerability scoring systems in a comparable and reproducible manner. It allows evaluators to investigate trade-offs between precision, consistency, and sensitivity, and to assess how design decisions influence prioritisation outcomes over time. As such, the methodology offers significant potential to support the refinement of existing scoring systems, guide parameter tuning, and inform the development of future vulnerability prioritisation models that are better aligned with organisational risk management objectives.

Chapter 6

Forthcoming Work and Conclusions

This work proposed a new method for comparing vulnerability scoring systems, based on distance metrics relative to an ideal ordering. By applying this method to evaluate SecScore, it was possible to demonstrate how the approach enables a rigorous and comparable assessment of effectiveness in vulnerability prioritisation. The evaluation confirmed that SecScore provides meaningful improvements over CVSS, particularly for Critical and High severity vulnerabilities, where prioritisation quality is most impactful for operational security.

Furthermore, the analysis of different time windows highlighted the influence that the choice of historical horizon can have on the quality of the generated scores. Short horizons may miss relevant exploitation patterns, while excessively long ones can dilute recent threat signals. This finding underscores the importance of carefully tuning the temporal scope when designing or applying scoring systems that rely on historical data.

The method presented therefore proves to be a versatile and robust tool for future studies on the evaluation and development of scoring systems. It offers a transparent way to benchmark different approaches, making it possible to assess not only whether a system is effective, but also under what conditions its performance is optimal.

As future work, we intend to experiment with different types of vulnerability features, in order to explore the impact these may have on the development of models adopted by scoring systems such as SecScore. Broadening the evaluation framework to incorporate additional features and alternative prioritisation logics will support the design of more adaptive, risk-aligned approaches to vulnerability management. Moreover, integrating environmental and contextual information — for example, through graph-based models of system components and dependencies — represents a promising path toward further improving the accuracy and adaptability of vulnerability prioritisation.

Bibliography

[1] Mario Angelelli, Serena Arima, Christian Catalano, and Enrico Ciavolino. A robust statistical framework for cyber-vulnerability prioritisation under partial information in threat intelligence. *Expert Syst. Appl.*, 255(PB), December 2024. doi:10.1016/j.eswa.2024.124572.

[2] TEAM ASCEND. The Five Stages of Vulnerability Management, 2019. Accessed: 2024-11-18. URL: https://blog.teamascend.com/stages-of-vulnerability-management.

[3] Artur Balsam, Maciej Nowak, Michal Walkowski, Jacek Oko, and Sławomir Sujecki. Comprehensive comparison between versions CVSS v2.0, CVSS v3.x and CVSS v4.0 as vulnerability severity measures. In *Proc. of the 2024 24th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, 2024. doi:10.1109/ICTON62926.2024.10647452.

[4] Artur Balsam, Michał Walkowski, Maciej Nowak, Jacek Oko, and Sławomir Sujecki. Automated calculation of CVSS v3.1 temporal score based on apache log4j 2021 vulnerabilities. *2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, page 1–3, September 2023. doi:10.23919/softcom58365.2023.10271671.

[5] Nick Cavalancia. Vulnerability management explained, 2020. Accessed: 2024-11-18. URL: https://levelblue.com/blogs/security-essentials/vulnerability-management-explained.

[6] Felipe Colombelli, Vítor Kehl Matter, Bruno Iochins Grisci, Leomar Lima, Karine Heinen, Marcio Borges, Sandro José Rigo, Jorge Luis Victória Barbosa, Rodrigo Da Rosa Righi, Cristiano André Da Costa, and Gabriel De Oliveira Ramos. Multi-objective prioritization for data center vulnerability remediation. In *Proc. of the 2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 01–08, 2022. doi:10.1109/CEC55065.2022.9870289.

[7] The MITRE Corporation. Common Vulnerabilities and Exposures (CVE), 2024. Accessed: 2024-11-18. URL: https://www.cve.org.

[8] National Vulnerability Database. National Vulnerability Database Vulnerability Metrics, 2024. Accessed: 2024-11-18. URL: https://nvd.nist.gov/vuln-metrics/cvss.

[9] National Vulnerability Database. NVD CVSS v4.0 Official Support, 2024. Accessed: 2024-11-18. URL: https://nvd.nist.gov/general/news/cvss-v4-0-official-support.

[10] Virender [Virender Dhiman] Dhiman. Automated vulnerability prioritization and remediation using deep learning. *JOURNAL OF BASIC SCIENCE AND ENGINEERING*, 20:86–97, August 2023. URL: https://yigkx.org.cn/index.php/jbse/article/view/303.

[11] Henry Howland. CVSS: ubiquitous and broken. *Digital Threats Research and Practice*, 4(1):1–12, October 2021. doi:10.1145/3491263.

[12] Sam Humphries. 4 Stages of Vulnerability Management: A Process for Risk Mitigation, 2024. Accessed: 2024-11-18. URL: https://www.exabeam.com/blog/infosec-trends/4-stages-of-vulnerability-management-a-process-for-risk-mitigation/.

[13] Kamil Imtiaz. Vulnerability Management Lifecycle, 2022. Accessed: 2025-11-27. URL: https://www.crowdstrike.com/en-us/cybersecurity-101/exposure-management/vulnerability-management-lifecycle/.

[14] Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. Exploit Prediction Scoring System (EPSS). *Digital Threats Research and Practice*, 2(3):1–17, 6 2021. doi:10.1145/3436242.

[15] Jay Jacobs, Sasha Romanosky, Octavian Suciu, Benjamin Edwards, and Armin Sarabi. Enhancing vulnerability prioritization: Data-driven exploit predictions with community-driven insights, 2023. URL: https://arxiv.org/abs/2302.14172, arXiv:2302.14172.

[16] Shaofeng Kai, Jinghua Zheng, Fan Shi, and Zhifan Lu. A CVSS-based vulnerability assessment method for reducing scoring error. In *Proc. of the 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*, pages 25–32, 2021. doi:10.1109/CECIT53797.2021.00013.

[17] Victor R. Kebande, Ivans Kigwana, H.S. Venter, Nickson M. Karie, and Ruth D. Wario. Cvss metric-based analysis, classification and assessment of computer network threats and vulnerabilities. In *2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, pages 1–10, 2018. doi:10.1109/ICABCD.2018.8465420.

[18] Viktoria Koscinski, Mark Nelson, Ahmet Okutan, Robert Falso, and Mehdi Mirakhorli. Conflicting scores, confusing signals: An empirical study of vulnerability scoring systems. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, page 1904–1918, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3719027.3765210.

[19] Matthew Kosinski. Vulnerability Management Process, 2024. Accessed: 2024-11-18. URL: https://www.ibm.com/think/topics/vulnerability-management.

[20] Matthew Kosinski. What is the vulnerability management lifecycle?, 2024. Accessed: 2025-11-27. URL: https://www.ibm.com/think/topics/vulnerability-management-lifecycle#:~:text=The%20vulnerability%20management%20lifecycle%20is,to%20the%20National%20Vulnerability%20Database.

[21] Qixu Liu and Yuqing Zhang. Vrss: A new system for rating and scoring vulnerabilities. *Comput. Commun.*, 34(3):264–273, March 2011. doi:10.1016/j.comcom.2010.04.006.

[22] Gordon "Fyodor" Lyon and the Nmap Project. Nmap: The Network Mapper, 2025. Accessed: 2025-09-04. URL: https://nmap.org/.

[23] Walid El Maouaki, Nouhaila Innan, Alberto Marchisio, Taoufik Said, Mohamed Bennai, and Muhammad Shafique. Quantum clustering for cybersecurity. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 02, pages 5–10, 2024. doi:10.1109/QCE60285.2024.10243.

[24] Takashi Minohara and Masaya Shimakawa. Security risk growth models for software vulnerability assessment. In *Proc. of the 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 32–35, 2023. doi:10.1109/DSN-W58399.2023.00026.

[25] Andrey Nikonov, Alexey Vulfin, Vladimir Vasilyev, Anastasia Kirillova, and Vladimir Mikhailov. System for estimation CVSS severity metrics of vulnerability based on text mining technology. In *Proc. of the 2021 International Conference on Information Technology and Nanotechnology (ITNT)*, pages 1–5, 2021. doi:10.1109/ITNT52450.2021.9649232.

[26] FIRST—Forum of Incident Response and Security Teams. A Complete Guide to the Common Vulnerability Scoring System, 2024. Accessed: 2025-11-26. URL: https://www.first.org/cvss/v2/guide.

[27] FIRST—Forum of Incident Response and Security Teams. Common Vulnerability Scoring System (CVSS), 2024. Accessed: 2024-11-18. URL: https://www.first.org/cvss/.

[28] FIRST—Forum of Incident Response and Security Teams. Common Vulnerability Scoring System version 3.1 Specification Document, 2024. Accessed: 2024-11-18. URL: https://www.first.org/cvss/v3.1/specification-document.

[29] FIRST—Forum of Incident Response and Security Teams. Common Vulnerability Scoring System version 4.0 Specification Document Version: 1.2, 2024. Accessed: 2024-11-18. URL: https://www.first.org/cvss/v4.0/specification-document.

[30] FIRST—Forum of Incident Response and Security Teams. Exploit Prediction Scoring System (EPSS), 2024. Accessed: 2024-11-18. URL: https://www.first.org/epss/model.

[31] Qualys. CVSS v4 Is Now Live and What You Need To Know About It, 2024. Accessed: 2024-11-18. URL: https://blog.qualys.com/product-tech/2023/11/02/cvss-v4-is-now-live-and-what-do-you-need-to-know.

[32] Rapid7. Vulnerability Management Process: Scanning, Prioritizing, and Remediating, 2024. Accessed: 2024-11-18. URL: https://www.rapid7.com/fundamentals/vulnerability-management-and-scanning/.

[33] Ali Raza and Waseem Ahmed. Threat and vulnerability management life cycle in operating systems: A systematic review. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, 9(1):15010–15013, 2022. JMESTN42353972. URL: https://www.jmest.org/wp-content/uploads/JMESTN42353972.pdf.

[34] Harri Renney, Isaac Chenchiah, Maxim Nethercott, Rohini Paligadu, and James Lang. An open and adaptable approach to vulnerability risk scoring. *Journal of Cyber Security*, 7:221–238, 07 2025. doi:10.32604/jcs.2025.064958.

[35] Miguel Santana, Vinicius V. Cogo, and Alan Oliveira de Sá. Secscore: Enhancing the CVSS threat metric group with empirical evidences, 2024. URL: https://arxiv.org/abs/2405.08539, arXiv:2405.08539.

[36] Alex Scroxton. 2024 seeing more CVEs than ever before, but few are weaponised, 2024. Accessed: 2024-11-18. URL: https://www.computerweekly.com/news/366600424/2024-seeing-more-CVEs-than-ever-before-but-few-are-weaponised.

[37] Ensar Seker and Weizhi Meng. Xvrs: Extended vulnerability risk scoring based on threat intelligence. In *Proc. of the 2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom)*, pages 516–523, 2023. doi:10.1109/MetaCom57706.2023.00094.

[38] Karl Smith. *Precalculus: A Functional Approach to Graphing and Problem Solving*. Jones & Bartlett Publishers, 2013.

[39] Jonathan Spring, Eric Hatleback, Allen Householder, Art Manion, and Deana Shick. Time to change the CVSS? *IEEE Security & Privacy*, 19(2):74–78, 2021. doi:10.1109/MSEC.2020.3044475.

[40] Matthias Studer and Gilbert Ritschard. A comparative review of sequence dissimilarity measures. *LIVES Working Papers*, 01 2014. doi:10.12682/lives.2296-1658.2014.33.

[41] Inc. Tenable. Nessus Vulnerability Scanner, 2025. Accessed: 2025-09-04. URL: https://www.tenable.com/products/nessus.

[42] Julia Wunder, Andreas Kurtz, Christian Eichenmüller, Freya Gassmann, and Zinaida Benenson. Shedding light on CVSS scoring inconsistencies: A user-centric study on evaluating widespread security vulnerabilities, 2024. URL: https://arxiv.org/abs/2308.15259, arXiv:2308.15259.

[43] Zhen Zeng, Dijiang Huang, Guoliang Xue, Yuli Deng, Neha Vadnere, and Liguang Xie. Illation: Improving vulnerability risk prioritization by learning from network. *IEEE Transactions on Dependable and Secure Computing*, 21(4):1890–1901, July 2023. doi:10.1109/tdsc.2023.3294433.

[44] Zhen Zeng, Zhun Yang, Dijiang Huang, and Chun-Jen Chung. Licality—likelihood and criticality: vulnerability risk prioritization through logical reasoning and deep learning. *IEEE Transactions on Network and Service Management*, 19(2):1746–1760, December 2021. doi:10.1109/tnsm.2021.3133811.

[45] Siqi Zhang, Minjie Cai, Mengyuan Zhang, Lianying Zhao, and Xavier de Carné de Carnavalet. The flaw within: Identifying CVSS score discrepancies in the nvd. In *Proc. of the 2023 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 185–192, 2023. doi:10.1109/CloudCom59040.2023.00039.