Computers & Security 132 (2023) 103382

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A framework for designing vulnerability metrics

Massimiliano Albanese*, Ibifubara Iganibo, Olutola Adebiyi

George Mason University, 4400 University Blvd, Fairfax, 22030, Virginia, USA

ARTICLE INFO

Article history:
Received 5 December 2022
Revised 23 June 2023
Accepted 4 July 2023
Available online 8 July 2023

Keywords:
Vulnerability analysis
Vulnerability graphs
Vulnerability metrics
Vulnerability scoring systems

ABSTRACT

Vulnerability analysis has long been used to evaluate the security posture of a system. Different approaches, including vulnerability graphs and various vulnerability metrics, have been used to study the vulnerability landscape and provide security analysts with cyber situational awareness. However, most current solutions still lack a principled approach to quantifying various dimensions of known vulnerabilities in a way that can easily adapt to different applicative domains and operating conditions. To address this limitation, we introduce a vulnerability metrics framework that extends and generalizes our previous metrics for evaluating the exploitation likelihood of a vulnerability and the exposure factor of system components to vulnerability exploits. We argue that the factors influencing these metrics and their relative weights depend on the specific applicative domain, defender's priorities, and attacker's knowledge. Thus, instead of providing a static set of equations, we establish a framework for instantiating the equations that best model the scenario being considered. We combine likelihood and exposure factor metrics into a severity score that allows us to rank vulnerabilities. In our evaluation, we demonstrate that ranking vulnerabilities solely based on their CVSS scores is not sufficient for effective prioritization, due to the limited number of possible distinct severity values compared to the sheer number of existing vulnerabilities. We define a ranking quality score and show that considering additional information about vulnerabilities helps refine their ranking, providing more actionable intelligence to security analysts.

1. Introduction

Vulnerability analysis has long been used to evaluate the security posture of systems and their ability to respond to multi-step attacks. Different approaches – including topological vulnerability analysis and a vast array of scoring and ranking systems – have been developed to study the properties of individual vulnerabilities and the relationships among multiple vulnerabilities, with the ultimate objective of providing security analysts with cyber situational awareness about the current vulnerability landscape (Albanese and Jajodia, 2018; Albanese et al., 2013; Ammann et al., 2002; Jajodia et al., 2005).

More recently, vulnerability graphs have been adopted as part of a multi-layer graph approach to configuration analysis and optimization, referred to as SCIBORG (Soroush et al., 2020). As cyber systems are becoming more complex and interconnected, configuration analytics and optimization are becoming increasingly critical for their correct and secure operation. Attackers usually rely on unpatched vulnerabilities and configuration errors to gain unauthorized access to system resources. Misconfiguration can occur at any level of a system's software architecture, and correctly configuring systems becomes more complex when many interconnected components are involved. In 2017, Security Misconfiguration was listed by OWASP amongst the ten most critical web application security risks (van der Stock et al., 2017).

However, in the most recent OWASP ranking of software risk, Misconfiguration moved higher in the ranking (van der Stock et al., 2021). SCIBORG's multi-faceted approach aims at modeling relationships between the components, configuration parameters, and vulnerabilities of a complex system by ingesting data from a number of different data sources, including but not limited to system documentation, operating procedures, and reports from vulnerability scanners. The resulting graph model is then analyzed with the goal of improving the security of the system while preserving its functionality. Correctly quantifying the risk represented by unpatched vulnerabilities is critical for this process to effectively improve a system's security.

Most current solutions still lack a principled approach to quantifying various dimensions of the problem in a way that can easily adapt the scoring and ranking of known vulnerabilities to different applicative domains and operating conditions. The Common Vulnerability Scoring System (CVSS) defines a Base score and allows security administrators to augment the Base score by using Temporal and Environmental metrics in order to yield a severity rat-

* Corresponding author.
E-mail address: malbanes@gmu.edu (M. Albanese).

https://doi.org/10.1016/j.cose.2023.103382
0167-4048/

ing that better aligns with their unique environment. Environmental metrics are specific to an organization and include attributes related to business criticality of the exposed asset, mitigations or compensating controls that are in place. However, administrators would be required to score each vulnerability with respect to each submetric based on their specific environment, which is time consuming and prone to errors and to the subjectivity of the administrators themselves. Additionally, as shown for the base metrics in Sections 6.1 and 6.2, the small number of discrete values that can be assigned to each submetric results in a limited number of distinct severity score values. To address this limitation, we introduce a vulnerability metrics framework that extends and generalizes our previous metrics for evaluating the exploitation likelihood of a vulnerability and the exposure factor of system components to vulnerability exploits (Iganibo et al., 2021). We argue that the factors influencing these metrics and their relative weights depend on the specific applicative domain, defender's priorities, and attacker's knowledge. Thus, instead of providing a static set of equations, we define a framework that allows its users to instantiate the equations that best model the scenario being considered. We combine likelihood and exposure factor metrics into a severity score that allows us to rank vulnerabilities. Our approach aims at allowing administrators to improve their ability to discriminate between vulnerabilities with very similar severity levels and to isolate those which pose the greatest risk to their organization. In our evaluation, we demonstrate that ranking vulnerabilities based on their CVSS scores alone is not sufficient to provide a high-quality ranking, as there may be thousands of vulnerabilities being assigned the same identical score. Instead, we show that considering more information about vulnerabilities helps refine their ranking, providing more actionable intelligence to security analysts.

In summary, the key contributions of this paper are: (i) a general framework for defining two key vulnerability metrics: exploitation likelihood and exposure factor; (ii) an inventory of variables that can potentially affect either of these two metrics; (iii) general principles for selecting variables to be considered in the computation of these two metrics, and (iv) a metric to evaluate the quality of the resulting vulnerability ranking. Overall, our framework aims at guiding security administrators in the definition of metrics that are suitable for the specific environment in which they are being deployed by giving them control over the data and corresponding variables that are used in the computation of the metrics and the relative weights of such variables. Differently from CVSS Environmental scores, administrators are not required to score each vulnerability with respect to a predefined set of submetrics, although any environmental submetric could in principle be used as a variable in our model, making it more general, flexible, and usable.

The remainder of the paper is organized as follows. Section 2 discusses related work, whereas Section 3 summarizes current standards in vulnerability scoring and provides a brief overview of IDS rules. Next, Section 4 gives an overview of the SCIBORG model introduced in (Soroush et al., 2020). Then, Section 5 introduces the proposed vulnerability metrics, and Section 6 reports on the results of our evaluation. Finally, Section 7 gives some concluding remarks and outlines possible future research directions.

2. Related work

The goal of vulnerability management is to effectively and intelligently prioritize remediation efforts based on actionable recommendations that consider both external variables and intrinsic features of existing vulnerabilities. It is important to understand the threat landscape in order to reduce the attack surface and remediate or mitigate the threats. Configuration analytics and optimization aim at reducing the attack surface, making it critical to measure a system's susceptibility to attacks. Over the years, researchers and practitioners have proposed various attack surface measurement techniques and metrics (Bopche et al., 2019; Manadhata and Wing, 2011; Stuckman and Purtilo, 2012; Yoon et al., 2020). However, these techniques do not typically consider the relationships between system components, vulnerabilities, configurations, and attacker's knowledge of the system. This limitation prevents existing metrics from accurately measuring a system's attack surface. More recently, research has been done to ascertain the importance of the information an adversary has about the system to enhance remediation and mitigation plans (Goohs et al., 2022). Remediation and mitigation efforts are resource-intensive and are subject to constraints in large networked systems. To adequately adopt countermeasures to attacks, an accurate assessment of the system's susceptibility is necessary. Furthermore, to develop comprehensive cyber situational awareness [8], and in line with more traditional risk analysis approaches, one has to distinguish between the likelihood that a vulnerability might be exploited and the impact a successful exploitation would cause. Current approaches to scoring and ranking of vulnerabilities are coarse-grained and do not consider the specific features of a system. A recent study argues that the severity score of a vulnerability as calculated by the Common Vulnerability Scoring System (CVSS) (Spring et al., 2021) does not directly reflect the risk represented by that vulnerability and how quickly one should respond to it. Furthermore, CVSS does not handle relationships between vulnerabilities. Different vulnerability metrics have varying levels of relevance depending on the specific context and application, therefore it is important to consider different attack models in defining metrics for guiding risk-based decisions.

There have been other research efforts focused on prioritization and remediation of vulnerabilities using data other than CVSS scores. One such effort resulted in the Exploit Prediction Scoring System (EPSS) (Jacobs et al., 2021), which is an open, data-driven approach for estimating the likelihood that a software vulnerability will be exploited in the wild, using current threat information from CVE and real-world exploit data. However, this model does not consider the impact of vulnerability exploits.

The approach we propose in this paper addresses these limitations by creating a general and customizable framework for defining vulnerability metrics, building upon existing literature on vulnerability graphs and vulnerability scoring.

3. Technical background

The approach to designing vulnerability metrics that we present in this article generalizes our previous work on vulnerability metrics (Iganibo et al., 2021), and relies on any vulnerability-specific information that is available to the users of the metrics framework, including data from the National Vulnerability Database (NVD), vulnerability scores computed through CVSS, and repositories of Intrusion Detection System (IDS) rules. In our previous work (Iganibo et al., 2021; 2022), we used vulnerability-level metrics to compute aggregate scores for weaknesses in the Common Weakness Enumeration (CWE) maintained by MITRE, and compared our CWE rankings against MITRE's CWE Top 25 Most Dangerous Software Weaknesses. The following subsections provide a brief overview of NVD, CVSS, CWSS, and IDS rules.

3.1. NVD

The National Vulnerability Database (NVD) is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP), and is maintained by the National Institute of Standards

and Technology (NIST). This data enables automation of vulnerability management, security measurement, and compliance.

NVD[1] is built upon and fully synchronized with the Common Vulnerabilities and Exposures (CVE) list of publicly known cybersecurity vulnerabilities. The CVE repository is maintained by MITRE and includes various details about each vulnerability – identification number, description, and public references. NVD augments it with severity scores, and impact ratings based on the Common Vulnerability Scoring System.

3.2. CVSS

As stated by the Forum of Incident Response and Security Teams (FIRST) – which currently maintains it – the Common Vulnerability Scoring System (CVSS) provides a means to *"capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity"*. This score is calculated based on three different metrics: (i) Base Score Metrics (required); (ii) Temporal Score Metrics (optional); and (iii) Environmental Score Metrics (optional).

CVSS is currently at version 3.1, but, in our studies, we normally use CVSS version 2.10, as base scores are available for a larger number of vulnerabilities. Temporal and Environmental metrics are defined to allow administrators to customize CVSS scores for their specific environment. Our model is not intended to replace CVSS but rather complement it. In fact, in our evaluation, we used the CVSS Exploitability and Impact scores as two variables in our model, and mentioned that any environmental metric, if defined, could be used as an additional variable. While the objective of our approach is in principle similar to the objective of the Environmental Metrics of CVSS, we achieve that objective in a different way, which can indeed be considered complementary to CVSS. Our approach relies on including any publicly or otherwise readily available data that can be mapped to individual CVEs (e.g., IDS rules, exploits) with minimal user effort, and it allows administrators to include new variables in the calculation of likelihood and exposure factor in order to take this data into account, whereas the CVSS approach relies on administrators assessing each vulnerability with respect to a predefined set of environmental submetrics (e.g., Modified Attack Vector (MAV), Modified Privileges Required (MPR)), which is a time-consuming and error-prone process. A study on the impact of environmental metrics on CVSS scores (Gallon, 2010) highlights the challenges posed by the Adjusted Impact score formula in accurately reflecting the distribution of vulnerabilities and the potential limitations it introduces for administrators seeking to discriminate vulnerabilities within their systems.

CVSS Base Score is computed through Eq. 1 below.

$$BaseScore = (0.6 \cdot I + 0.4 \cdot E - 1.5) \cdot f(I) \tag{1}$$

where I and E are the Impact and Exploitability scores defined by Eqs. 2 and 3 respectively, and $f(I)$ is defined by Eq. 4. The Impact score quantifies the consequences of an exploit, whereas the Exploitability score captures how easy to exploit a vulnerability is.

$$I = 10.41 \cdot \left(1 - (1 - I_C) \cdot (1 - I_I) \cdot (1 - I_A)\right) \tag{2}$$

$$E = 20 \cdot AC \cdot A \cdot AV \tag{3}$$

$$f(I) = \begin{cases} 0, & \text{if } I = 0 \\ 1.176, & \text{otherwise} \end{cases} \tag{4}$$

The I_C, I_I, and I_A scores in Eq. 2 are the confidentiality, integrity, and availability impact respectively, as defined in Table 1.

[1] https://nvd.nist.gov/

Table 1
Impact metrics.

	Confidentiality Impact (I_C)	Integrity Impact (I_I)	Availability Impact (I_A)
None	0.000	0.000	0.000
Partial	0.275	0.275	0.275
Complete	0.660	0.660	0.660

Table 2
Exploitability metrics.

Access Compl. (AC)		Authentication (A)		Access Vector (AV)	
High	0.35	Multiple	0.450	Local	0.395
Medium	0.61	Single	0.560	Adjacent	0.646
Low	0.71	None	0.704	Network	1.000

The AC, A, and AV scores in Eq. 3 are the exploitability metrics Access Complexity, Authentication, and Access Vector, as defined in Table 2.

It is important to note that, since all the submetrics involved in their computation can assume one of only a few discrete values, the Impact and Exploitability scores will also have one of a limited number of discrete values. Thus, ranking thousands of vulnerabilities based on their CVSS scores is unpractical, as we will demonstrate later in Section 6.

3.3. CWE and CWSS

Common Weakness Enumeration (CWE) is a system that provides a structured list of clearly defined software and hardware weaknesses[2]. A software weakness is not necessarily a vulnerability, but weaknesses may become vulnerabilities. MITRE's Common Weakness Scoring System (CWSS) provides a mechanism for prioritizing software weaknesses that are present within software applications in a consistent and flexible manner[3]. It is a collaborative, community-based effort that is addressing the needs of its stakeholders across government, academia, and industry.

CWSS is organized into three metric groups: Base Finding, Attack Surface, and Environmental. Each group includes multiple metrics – also known as factors – that are used to compute a CWSS score for a weakness. While discussing the formulation of these metrics goes beyond the scope of this paper and we refer the reader to the documentation for further details, in the following we describe the method MITRE used to rank the most dangerous weaknesses: in our previous work (Iganibo et al., 2021), we relied on this approach to validate our metrics, as explained later in Section 6.

Eq. 5 defines the set of CVEs mapped to a given CWE, and Eq. 6 defines the number of times each CWE is mapped to CVE entries[4]

$$C(CWE_i) = \{CVE_j \in NVD, CVE_j \rightarrow CWE_j\} \tag{5}$$

$$Freqs = \{|C(CWE_i)|, CWE_i \in NVD\} \tag{6}$$

Then Eqs. 7 and 8 respectively compute the frequency and severity of a CWE, where the severity is based on the average CVSS score. Both the frequency and severity are normalized between 0 and 1.

$$Fr(CWE_i) = \frac{|C(CWE_i)| - min(Freq)}{max(Freq) - min(Freq)} \tag{7}$$

[2] https://cwe.mitre.org/

[3] https://cwe.mitre.org/cwss/

[4] We slightly abuse notation and use $CWE_i \in NVD$ to denote a CWE that is mapped to at least one CVE entry in NVD.

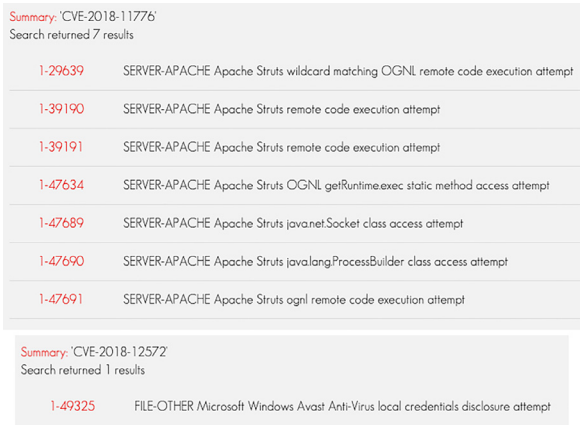

Fig. 1. Snort rules associate with different CVEs.

$$Sv(CWE_i) = \frac{avg_{CWE_i}(CVSS) - min(CVSS)}{max(CVSS) - min(CVSS)} \tag{8}$$

Finally Eq. 9 defines the overall score of a CWE as the product of its frequency and severity, normalized between 0 and 100.

$$Score(CWE_i) = Fr(CWE_i) \cdot Sv(CWE_i) \cdot 100 \tag{9}$$

3.4. IDS rules

In our approach, we use Intrusion Detection System (IDS) rules as one of the factors influencing the computation of both the likelihood of a vulnerability exploit and the exposure factor of a system component to a vulnerability. However, we distinguish between *known* and *deployed* rules, and only consider rules that are explicitly mapped to CVE entries. We use the term *known IDS rule* to refer to any IDS rule that is available to the community through publicly accessible repositories. Our assumption is that the existence of known IDS rules associated with a given vulnerability may decrease the likelihood of exploiting that vulnerability, as an attacker may prefer to target vulnerabilities that can be exploited without triggering IDS alerts. As an example, Fig. 1 shows the results of searching the Snort rule repository for two different CVEs, that is CVE-2018-11776 and CVE-2018-12572 respectively.

Instead, we use the term *deployed IDS rule* to refer to any IDS rule that is being actively used by a deployed IDS. Deployed rules may include a subset of known rules or ad hoc rules developed by the system's administrators. An attacker may not be aware of what IDS rules are actually in use, but early detection of intrusions may help mitigate the consequences of an exploit, therefore we take deployed rules into account in the computation of exposure factors.

While our approach is general and does not require the adoption of a specific intrusion detection system, we have used Snort[5] and Suricata[6].

4. The SCIBORG model

This section gives a brief overview of the SCIBORG graph model that was presented in (Soroush et al., 2020) and that has offered an ideal case study for the vulnerability metrics we have developed. The metrics framework we introduce in this paper is designed to address current limitations of this model and similar models by introducing the capability of quantifying two different dimensions of

[5] https://www.snort.org/
[6] https://suricata-ids.org/

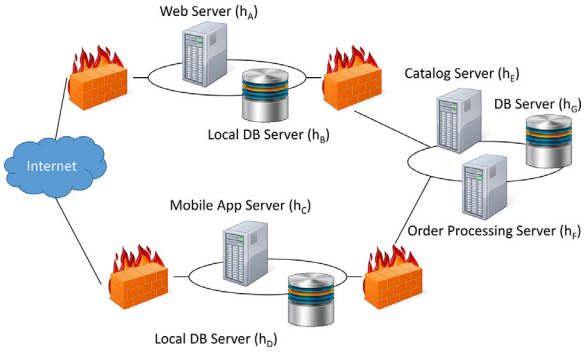

Fig. 2. Network diagram of a notional distributed system.

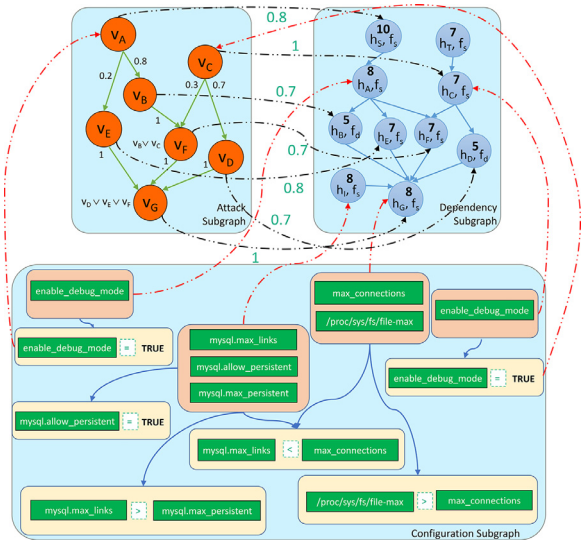

Fig. 3. The SCIBORG graph for the system of Fig. 2.

the vulnerabilities being considered, which in turn may be leveraged to define higher level metrics, such as attack surface metrics. It is important to note that our approach is not limited to graphs generated by SCIBORG. In fact, it is applicable to any graph model that relies on vulnerability graphs.

SCIBORG's approach is based on modeling a distributed system as a three-layer directed graph encoding all the information needed to reason upon the optimality of system configurations. The three layers are (i) a dependency subgraph; (ii) a vulnerability subgraph; and (iii) a configuration subgraph. For illustrative purposes, a three-layer graph corresponding to the notional distributed system of Fig. 2 is depicted in Fig. 3. The following subsections describe in detail the first two sub graphs, which are the most relevant to the scope of this paper, and we refer the reader to our previous work (Soroush et al., 2020) for details about the third subgraph.

4.1. Dependency subgraph

Failures of one component can impact the security and functionality of other components. Therefore, globally optimal security decisions need to rely on dependency information. Several approaches have been proposed to discover implicit or undocumented dependencies (Bahl et al., 2006; Natrajan et al., 2012).

A node in the dependency subgraph represents a system component (host, service, etc.), and a directed edge represents a dependency between two components. When dependencies are accurately captured, dependency graphs are expected to be acyclic. To

$$\mathcal{X}_l^{\uparrow} = \{X \in \mathcal{X}_l \mid (\forall v_1, v_2 \in V)\,((X(v_1) \leq X(v_2) \wedge ((\forall X' \in \mathcal{X} \setminus \{X\})\,(X(v_1) = X(v_2)))) \implies \rho(v_1) \leq \rho(v_2))\} \quad (10)$$

$$\mathcal{X}_l^{\downarrow} = \{X \in \mathcal{X}_l \mid (\forall v_1, v_2 \in V)\,((X(v_1) \leq X(v_2) \wedge ((\forall X' \in \mathcal{X} \setminus \{X\})\,(X(v_1) = X(v_2)))) \implies \rho(v_1) \geq \rho(v_2))\} \quad (11)$$

$$\mathcal{X}_e^{\uparrow} = \{X \in \mathcal{X}_e \mid (\forall v_1, v_2 \in V)\,((X(v_1) \leq X(v_2) \wedge ((\forall X' \in \mathcal{X} \setminus \{X\})\,(X(v_1) = X(v_2)))) \implies ef(v_1) \leq ef(v_2))\} \quad (12)$$

$$\mathcal{X}_e^{\downarrow} = \{X \in \mathcal{X}_e \mid (\forall v_1, v_2 \in V)\,((X(v_1) \leq X(v_2) \wedge ((\forall X' \in \mathcal{X} \setminus \{X\})\,(X(v_1) = X(v_2)))) \implies ef(v_1) \geq ef(v_2))\} \quad (13)$$

Fig. 4. Equations defining the set of variables $\mathcal{X}_l^{\uparrow}$, $\mathcal{X}_l^{\downarrow}$, $\mathcal{X}_e^{\uparrow}$, and $\mathcal{X}_e^{\downarrow}$ introduced in Section 5.2.

capture a wide range of relationships between components, each dependency is modeled as a function of the form $f : [0, 1]^n \rightarrow [0, 1]$, with $f(0, \ldots, 0) = 0$ and $f(1, \ldots, 1) = 1$.

Each component has an intrinsic utility for the owning organization and its *dependency function* defines its ability to provide its expected utility, based on the status of the components it depends on: the arguments of this function are the percentage residual utilities of those components and are in turn computed through each component's respective dependency function. A dependency function returns 1 when the component can provide 100% of its utility, and 0 when it has been completely compromised. Three types of dependency relationships are identified in (Soroush et al., 2020), namely *redundancy* (f_r), *strict dependence* (f_s), and *graceful degradation* (f_d), but such classification is not intended to be exhaustive, and other dependency relationships can be defined.

Fig. 3 includes the dependency subgraph for our notional system. An edge from h_A to h_B denotes that h_A depends on h_B. Each component node is labeled with the type of dependency and its utility. Utility values can be assigned by a domain expert or automatically derived by computing graph-theoretic centrality metrics (Kourtellis et al., 2015). In the security field, ad-hoc centrality measures have been used for botnet detection and mitigation (Venkatesan et al., 2015).

4.2. Vulnerability subgraph

Vulnerability subgraphs are powerful conceptual tools to represent knowledge about vulnerabilities and their dependencies. SCIBORG adopts a broader definition of *vulnerability*, but for the purpose of the analysis presented in this paper we refer to vulnerability graphs as formalized in Albanese and Jajodia (2018). A node in the vulnerability subgraph represents a known vulnerability, and an edge between two vulnerabilities, referred to as an ENABLES edge, indicates that exploiting the first vulnerability creates the preconditions to exploit the second one. An edge from a node in the vulnerability subgraph to a node in the dependency subgraph, referred to as a DEGRADES edge, indicates that the exploitation of a given vulnerability can impact a component to an extent quantified by the exposure factor labeling that edge.

The vulnerability subgraph for our notional system is also depicted in Figure. 3. This graph can be generated by combining information from network scanners (e.g., Nessus) and vulnerability databases (e.g., CVE, NVD), as shown in Ammann et al. (2002); Jajodia et al. (2005). The edges in the vulnerability subgraph of Fig. 3 are labeled with probabilities, which can be used to infer the most likely paths that an attacker might take in a multi-step attack. Determining these probabilities is an open research problem that we address in this paper, though useful heuristics exist (Albanese and Jajodia, 2018; Albanese et al., 2013), based on the assumption that vulnerabilities that require more resources, time, and skill are less likely to be exploited.

5. Vulnerability metrics

In this section, we formally define the proposed vulnerability metrics framework, which was designed to address the limitations of current approaches we discussed earlier. We argue that each vulnerability is defined by its *exploitation likelihood* and its *exposure factor* and that each of these two metrics is influenced by a number of different variables. Furthermore, the specific set of variables that influence these metrics and their relative weights may vary across application domains. In the following, we first provide an intuitive definition of the two metrics, then model the set of variables potentially affecting their values, and finally provide a formal mathematical formulation of the two metrics.

5.1. Exploitation likelihood and exposure factor

A vulnerability's susceptibility to becoming a target for exploitation by malicious users depends on a number of variables, including features of the vulnerability itself and characteristics of potential attackers. In our previous work, we only considered intrinsic features of vulnerabilities, as the problem of studying an attacker's skills and resources appeared to be an orthogonal problem. In fact, we were interested in scoring and comparing individual vulnerabilities for a fixed attack model. Several approaches have been proposed in the literature to study how the skills and resources available to different types of attackers impact their ability to compromise a target system (Leversage and Byres, 2008). To build upon our previous work and available literature, we argue that, while we continue to use variables capturing intrinsic features of the vulnerabilities, the choice of variables to consider and their relative weights can be used to model different types of attackers, ranging from adversaries who are only aware of a vulnerability's CVSS scores to adversaries that can perform reconnaissance on the target systems and discover unpatched vulnerabilities.

We define the *exploitation likelihood* (or simply *likelihood*) $\rho(v)$ of a vulnerability v as the probability that an attacker will attempt to exploit that vulnerability, if given the opportunity. An attacker has the *opportunity* to exploit a vulnerability if certain preconditions are met, most notably if they have access to the vulnerable host. Specific preconditions may vary depending on the specific characteristics of each vulnerability, as certain configuration settings may prevent access to vulnerable portions of the target software.

We define the *exposure factor* as the relative loss of utility of an asset due to a vulnerability exploit. The term is borrowed from classic risk analysis terminology, where the exposure factor (EF) represents the relative damage that an undesirable event – a cyber attack in our case – would cause to the affected asset. The single loss expectancy (SLE) of such an incident is then computed as the product between its exposure factor and the asset value (AV), that is $SLE = EF \times AV$.

5.2. Variables affecting the metrics

Let V denote the set of all known vulnerabilities, $\mathcal{X}_l$ denote a set of variables that influence the exploitation likelihood, and $\mathcal{X}_e$ denote a set of variables that influence the exposure factor. We define $\mathcal{X}_l^{\uparrow}$ and $\mathcal{X}_l^{\downarrow}$ as the sets of variables that respectively contribute to increasing and decreasing the likelihood as their values increase. $\mathcal{X}_l^{\uparrow}$ and $\mathcal{X}_l^{\downarrow}$ are defined by Equations 10 and 11 respectively. Similarly, we define $\mathcal{X}_e^{\uparrow}$ and $\mathcal{X}_e^{\downarrow}$ as the sets of variables that respectively contribute to increasing and decreasing the exposure as their values increase. $\mathcal{X}_e^{\uparrow}$ and $\mathcal{X}_e^{\downarrow}$ are defined by Equations (12) and (13) respectively.

Examples of variables in $\mathcal{X}_l^{\uparrow}$ include, but are not limited to, a vulnerability's exploitability score as captured by CVSS, the time since details about the vulnerability were published, and the set of known exploits.

CVSS Exploitability score. The CVSS Exploitability score captures how easy it is to exploit a vulnerability, based on different features captured by various sub-metrics, most notably Access Vector (AV) and Access Complexity (AC)[7] The Access Vector metric reflects the context in which a vulnerability can be exploited. Its value is higher for vulnerabilities that can be exploited remotely, and are therefore more likely to be exploited as the number of potential attackers is larger than the number of potential attackers that could exploit a vulnerability requiring physical access to the vulnerable host. The Attack Complexity metric reflects the amount of effort and resources required for a successful attack. Its value is higher for exploits that require little or no effort, and are therefore more likely to be exploited.

Time since publication. The time passed since details about the vulnerability were made public also plays a role in determining the likelihood of exploitation. In fact, the longer a vulnerability has been known, the more exploits may have been developed by the hacker community. While it is true that the likelihood that patches have been developed also increases with time, it is well-known that patches are not applied promptly and consistently across systems, thus giving attackers a window of opportunity to target known but unpatched vulnerabilities.

Vulnerability Exploits. The availability of exploits and Proofs of Concept (PoCs) associated with a vulnerability – which can be found in the Exploit-DB repository[8] or on MITRE's website, where CVEs are mapped to their exploits[9] – can provide an incentive for attackers to exploit specific vulnerabilities.

Examples of variables in $\mathcal{X}_l^{\downarrow}$ include, but are not limited to, the set of known Intrusion Detection System (IDS) rules associated with a vulnerability and the set of vulnerability scanning plugins.

Known IDS rules. Known IDS rules may influence the attacker's choice of vulnerabilities to exploit. With systems typically exposing multiple vulnerabilities, attackers may choose to avoid exploits that are more easily detectable.

Vulnerability scanning plugins. Vulnerability scanning tools can provide an inventory of existing system vulnerabilities. The availability of plugins to confirm the existence of a given vulnerability may make such vulnerability less likely to be exploited because attackers my expect that defenders would use such detection capabilities to detect and mitigate that vulnerability. Tenable Research designs plugins written in Nessus Attack Scripting Language (NASL) to detect vulnerabilities as they are discovered and released into the general public domain. These plugins contain information about the vulnerability, set of remediation actions, and the algorithm to test for the presence of the security issue[10].

Examples of variables in $\mathcal{X}_e^{\uparrow}$ include, but are not limited to, a vulnerability's impact score as captured by CVSS.

CVSS Impact score. The CVSS Impact score captures the impact of a vulnerability exploit on confidentiality, integrity, and availability.

Examples of variables in $\mathcal{X}_e^{\downarrow}$ include, but are not limited to, the set of deployed Intrusion Detection System (IDS) rules associated with a vulnerability.

Deployed IDS rules. IDS rules that are deployed on a system and actively monitoring for intrusions, can mitigate the consequences of an exploit through timely detection.

The list of variables presented here is not intended to be exhaustive and other variables could be identified and used in the calculation of both the exploitation likelihood and the exposure factor. For instance, it has been shown that the likelihood of exploiting a vulnerability also depends on the position of the vulnerable system within an attack path (Albanese and Jajodia, 2018). In fact, a vulnerability on a perimeter network may be more likely to be exploited than the same vulnerability on an internal network. Additionally, vulnerabilities that have similar characteristics as those an attacker has already exploited might be more easily exploited compared to completely different vulnerabilities. However, considering these situational variables would require to extend the model in order to consider each instance of a vulnerability as a separate entity for the purpose of scoring and ranking. Thus it is beyond the scope of the work presented here, but part of our future research plans.

5.3. Metrics definition

We formalize the exploitation likelihood as a function $\rho : V \to [0, 1]$ defined by Equation 14.

$$\rho(v) = \frac{\prod_{X \in \mathcal{X}_l^{\uparrow}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X \in \mathcal{X}_l^{\downarrow}} e^{\beta_X \cdot f_X(X(v))}} \tag{14}$$

Each variable contributes to the overall likelihood as a multiplicative factor between 0 and 1 that is formulated to account for *diminishing returns*. Factors corresponding to variables in $\mathcal{X}_l^{\uparrow}$ are of the form $1 - e^{-\alpha_X \cdot f_X(X(v))}$, where X is the variable, α_X is a tunable parameter, $X(v)$ is the value of X for v, and f_X is a monotonically increasing function used to convert values of X to scalar values, i.e., $x_1 < x_2 \Rightarrow f_X(x_1) \leq f_X(x_2)$. Similarly, factors corresponding to variables in $\mathcal{X}_l^{\downarrow}$ are of the form $\frac{1}{e^{\beta_X \cdot f_X(X(v))}} = e^{-\beta_X \cdot f_X(X(v))}$. We assume that each product evaluates to 1 when the corresponding set of variables is empty, i.e., $\prod_{X \in \mathcal{X}} (\ldots) = 1$ when $\mathcal{X} = \emptyset$.

The definition of the function f_X implies that the domain of each variable is a totally ordered set. While not every domain may be a totally order set, it is possible to map elements of the domain to elements of a totally ordered set. For instance, if the values of a variable are sets of objects, their respective cardinalities are totally ordered. If the function mapping the values of a variable X to values of a totally ordered set is a scalar function, then it can be used as the function f_X in Eq. 14. In most cases, when the values of X are already scalar values, we can define f_X as the identity function $f_X(x) = x$, but in the case of the time t since the vulnerability was disclosed, we use $f_X(t) = \sqrt{t}$ to model a less-than-linear relationship, as suggested by Tripwire[11].

Example 1 (Likelihood)**.** Consider a scenario in which an attacker's primary goal is to avoid detection. In this case, the choice of

[7] Access Vector (AV) and Access Complexity (AC) are common across CVSS 2 and CVSS 3, whereas other exploitability metrics are specific to either version.

[8] https://www.exploit-db.com/

[9] https://cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html

[10] https://www.tenable.com/plugins/families/about

[11] https://www.tripwire.com/solutions/vulnerability-and-risk-management

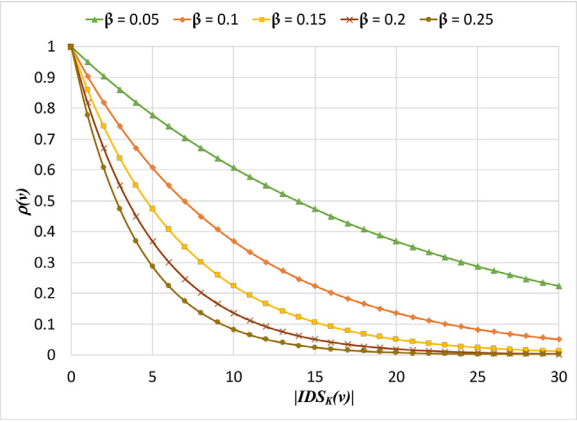

Fig. 5. Effect of variable $IDS_k(v)$ on the likelihood.

vulnerabilities to exploit is driven by the scarcity or complete lack of corresponding IDS rules. Thus, we can assume $\mathcal{X}_l^{\uparrow} = \emptyset$, $\mathcal{X}_l^{\downarrow} = \{IDS_k\}$, with $f_X(IDS_k(v)) = |IDS_k(v)|$, and instantiate Equation 14 as follows.

$$\rho(v) = \frac{1}{e^{\beta \cdot |IDS_k(v)|}} \tag{15}$$

Fig. 5 shows the effect of variable $IDS_k(v)$ on the likelihood for various values of the parameter β. As an example, for $\beta = 0.25$, the existence of 3 known IDS rules associated with a vulnerability v cuts the likelihood of exploiting v approximately in half.

The likelihood metric can be used to compute the probabilities associated with the edges of a vulnerability graph. When attempting to penetrate a complex networked system, attackers usually engage in multi-step attacks, which can be modeled through vulnerability graphs, as described in Section 4.2. At every step of the attack, adversaries will typically be able to choose among several vulnerabilities to exploit next in order to advance the attack. Therefore, for each node in the vulnerability subgraph, we need to compute a probability distribution over the outgoing ENABLES edges. As all the variables that can influence the attacker's choice of vulnerabilities to exploit have been factored into each vulnerability's likelihood, this probability distribution can be computed by normalizing the likelihood values of the *enabled* vulnerabilities and using the normalized values to label the corresponding ENABLES edges. Therefore, given an ENABLES edge $e = (u, v)$ (shown in Fig. 6), the probability of exploiting v after u is given by

$$\Pr(e) = \frac{\rho(v)}{\sum_{v' \, s.t. \, (u,v') \in E} \rho(v')} \tag{16}$$

We now formalize the exposure factor as a function $ef : V \to [0, 1]$ defined by Eq. 17.

$$ef(v) = \frac{\prod_{X \in \mathcal{X}_e^{\uparrow}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X \in \mathcal{X}_e^{\downarrow}} e^{\beta_X \cdot f_X(X(v))}} \tag{17}$$

Similar to the likelihood, each variable contributes to the exposure factor as a multiplicative factor between 0 and 1 that accounts for diminishing returns. Factors corresponding to variables in $\mathcal{X}_e^{\uparrow}$ are of the form $1 - e^{-\alpha_X \cdot f_X(X(v))}$, and factors corresponding to variables in $\mathcal{X}_e^{\downarrow}$ are of the form $\frac{1}{e^{\beta_X \cdot f_X(X(v))}} = e^{-\beta_X \cdot f_X(X(v))}$. Again, we assume that each product evaluates to 1 when the corresponding set of variables is empty, i.e., $\prod_{X \in \mathcal{X}}(\ldots) = 1$ when $\mathcal{X} = \emptyset$.

6. Experimental evaluation

This section describes the experiments we conducted to evaluate the proposed framework. In our previous work (Iganibo et al., 2021), we validated our approach by aggregating vulnerability-level metrics into a Common Weakness Enumeration (CWE) score for each CWE category and compared our ranking of CWEs against MITRE's CWE Top 25 Most Dangerous Software Weaknesses[12]. The results indicated that, when our framework is tuned to reproduce MITRE's experimental setting as closely as possible, the correlation between our ranking and MITRE's ranking is between 80% and 90%. Having already validated the general principles of our approach, this section will focus on reporting the results of experiments in various attack and defense scenarios. In each scenario, we make assumptions about the information available to the attacker – which results in the choice of variables to be included in $\mathcal{X}_l^{\uparrow}$ and $\mathcal{X}_l^{\downarrow}$ – and the information available to the defender – which results in the choice of variables to be included in $\mathcal{X}_e^{\uparrow}$ and $\mathcal{X}_e^{\downarrow}$. We assume that any information that is available to the attacker is also available to the defender, but not all information that is available to the defender is also available to the attacker. For instance, both the attacker and the defender are aware of known IDS rules associated with a vulnerability, but only the defender knows which rules are actually deployed within their systems.

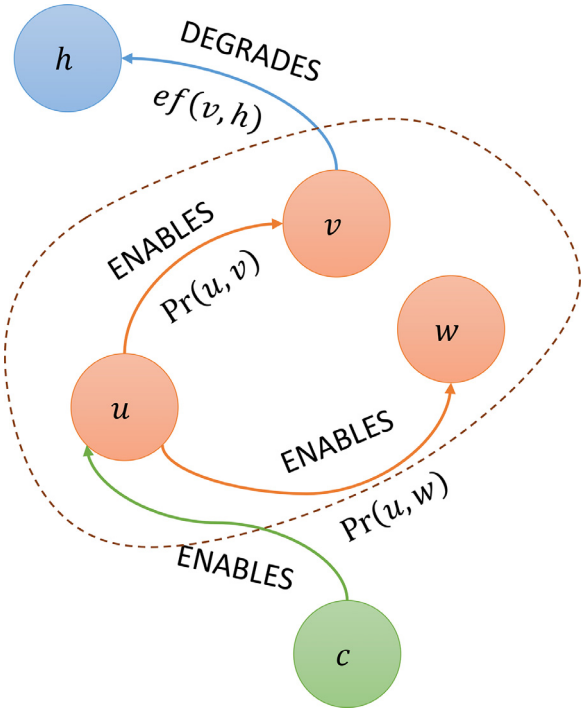

Fig. 6. Example of edges in the SCIBORG graph model.

The goal of the experiments described here is to understand how considering different combinations of variables leads to different rankings of vulnerabilities and how increasing the number of variables considered leads to more fine-grained rankings and an improved ability to discriminate between different vulnerabilities while prioritizing mitigation and remediation. The goal is not to directly compare our approach to CVSS, but rather to augment rankings based on CVSS Base scores alone by factoring in more variables, which may include, but are not limited to, CVSS Environmental scores, thus making our approach more general and flexible.

In our evaluation, we considered CVEs from 2020 and 2021, for a total of 33,741 vulnerabilities as of October 22, 2022, and ranked

[12] https://cwe.mitre.org/top25/

Table 3
Variables considered in each scenario.

	$\mathcal{X}_l^{\uparrow}$	$\mathcal{X}_l^{\downarrow}$	$\mathcal{X}_e^{\uparrow}$
Scenario 1	{CVSS_Exploitability}	∅	∅
Scenario 2	∅	∅	{CVSS_Impact}
Scenario 3	{CVSS_Exploitability}	∅	{CVSS_Impact}
Scenario 5	{CVSS_Exploitability},{Vuln_Exploit}	{Known_IDS_Rules}	{CVSS_Impact}
Scenario 6	{CVSS_Exploitability},{Vuln_Exploit}	{Known_IDS_Rules},{Vuln_Plugins}	{CVSS_Impact}
Scenario 7	{CVSS_Exploitability},{Age}	{Known_IDS_Rules}	{CVSS_Impact}

them based on the severity score defined by Eq. 18 below.

$$s(v) = \rho(v) \cdot ef(v) \tag{18}$$

We considered 7 different attack-defense scenarios, with each scenario resulting in a different choice of variables for $\mathcal{X}_l^{\uparrow}$, $\mathcal{X}_l^{\downarrow}$, $\mathcal{X}_e^{\uparrow}$, and $\mathcal{X}_e^{\downarrow}$. Table 3 shows the variables considered in each scenario. As a baseline, we considered only one variable in each of scenarios 1 and 2, and more variables in the remaining scenarios. Note that all the variables considered in the 7 scenarios of Table 3 capture publicly available scores and vulnerability data. To customize the rankings for their specific environment, administrators can add variables that capture environment-specific information, such as the sets of IDS rules actually deployed across the system.

In the following, we use the term r-ranking to denote a ranking comprising the top r distinct values of the severity score and showing the number of CVEs that are assigned that particular score. For each scenario, and for each rank r, we first compute how much the r-ranking deviates from the ideal scenario. The ideal scenario can be described as a ranking in which all the r most severe vulnerabilities have different severity scores (i.e., there are no ties between vulnerabilities).

$$\delta(r) = \sqrt{\frac{\sum_{i=1}^{r} \left(|CVE(r)| - 1\right)^2}{r}} \tag{19}$$

Then, we compute a quality score $Q(r)$ using the following equation.

$$Q(r) = e^{-\gamma \cdot \delta(r)} \tag{20}$$

where γ is a tunable parameter. This score is higher when the ranking is closer to the ideal scenario, with $Q(r) = 1$ when each vulnerability has a different score. The score decreases when there are more vulnerabilities with the same score or the number of vulnerabilities per rank is less uniform.

6.1. Scenario 1

In this scenario, we considered the CVSS Exploitability score as the only variable in the set $\mathcal{X}_l^{\uparrow}$ defined by Equation 10. No variables were considered for the other three sets, $\mathcal{X}_l^{\downarrow}$, $\mathcal{X}_e^{\uparrow}$, and $\mathcal{X}_e^{\downarrow}$. Equation 14 and 17 can be rewritten as follows for scenario 1.

$$\rho(v) = \prod_{X \in \{\text{CVSS\_Exploitability}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

$$ef(v) = 1$$

Table 4 shows the results for Scenario 1. For each rank r, the table shows the value of the severity score, the number of CVEs with that severity score, the cumulative number of vulnerabilities with that or higher severity score, the standard deviation, and the quality score of the partial ranking ending at r. In an ideal scenario, when vulnerabilities are ranked there should be no ties between CVEs, i.e., each CVE should have its unique score. Instead, the table shows that, for each of the top r distinct values of the severity score, there are multiple vulnerabilities (even thousands) with the same score. As shown in the table, the cumulative number of CVEs with one of the top 10 values of the severity score is 32,796 out of the 33,741 we considered, which makes this ranking not useful in practice.

Table 4
Ranking of CVEs in Scenario 1.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.91792	9308	9,308	9307.0000	0.00%
2	0.88352	9570	18,878	9438.9091	0.00%
3	0.86466	4683	23,561	8167.1524	0.00%
4	0.81732	3098	26,659	7240.4859	0.00%
5	0.80309	550	27,209	6480.7398	0.00%
6	0.74716	99	27,308	5916.2143	0.00%
7	0.72057	181	27,489	5477.7726	0.00%
8	0.70624	255	27,744	5124.7739	0.00%
9	0.66713	25	27,769	4831.6898	0.00%
10	0.62281	5027	32,796	4851.4711	0.00%

The quality score defined by Eq. 20 helps determine how much the ranking deviates from the ideal scenario. This score is an exponential function of the standard deviation between the numbers of vulnerabilities for each rank and a vector of 1′s corresponding to the ideal scenario (see Eq. 19). The quality score goes asymptotically to 0 as the standard deviation increases. Scenario 1 shows a high number of CVEs having the same severity score. This results in a high standard deviation and consequently in a virtually 0 quality score. Intuitively this ranking does not provide significant help for security administrators to make informed decisions when it comes to prioritizing vulnerability remediation. These results can be explained by examining Eq. 3, which defines the exploitability as a function of three variables, each of which can have only 3 possible values, resulting in a maximum of 27 possible values.

6.2. Scenario 2

In this scenario, we considered the CVSS Impact score as the only variable in the set $\mathcal{X}_e^{\uparrow}$ defined by Equation 10. No variables were considered for the other three sets, $\mathcal{X}_l^{\uparrow}$, $\mathcal{X}_l^{\downarrow}$, and $\mathcal{X}_e^{\downarrow}$. Equation 14 and 17 can be rewritten as follows for scenario 2.

$$\rho(v) = 1$$

$$ef(v) = \prod_{X \in \{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 5, the quality of the resulting ranking is again practically 0, due to the high number of vulnerabilities with the same score. These results can be explained by examining Eq. 2, which defines the impact as a function of three variables, each of which can have only the same 3 possible values, resulting in a maximum of 10 possible values (number of combination with repetition). Thus, the ability of a metric based solely on the CVSS impact score to discriminate among different vulnerabilities is even less than the metric used in the previous scenario, as confirmed

Table 5
Ranking of CVEs in Scenario 2.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.91792	4150	4150	4149.0000	0.00%
2	0.90699	7	4157	2933.7891	0.00%
3	0.89974	112	4269	2396.2859	0.00%
4	0.88057	46	4315	2075.3664	0.00%
5	0.85773	38	4353	1856.3379	0.00%
6	0.82183	885	5238	1732.5996	0.00%
7	0.79810	10,596	15,834	4313.8553	0.00%
8	0.70624	2034	17,868	4098.7578	0.00%
9	0.51568	15,873	33,741	6551.6657	0.00%

Table 6
Ranking of CVEs in Scenario 3.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.84257	867	867	866.0000	0.02%
2	0.82589	35	902	612.8262	0.22%
3	0.81099	717	1619	649.0424	0.15%
4	0.80829	10	1629	562.1052	0.36%
5	0.79494	7	1636	502.7693	0.66%
6	0.79369	745	2381	550.3667	0.41%
7	0.78732	10	2391	509.5519	0.61%
8	0.78424	2	2393	476.6423	0.85%
9	0.77799	11	2404	449.3950	1.12%
10	0.77797	26	2430	426.4068	1.41%

by the fact that all 33,741 vulnerabilities considered are assigned one of only 9 different scores.

6.3. Scenario 3

In this scenario, we considered the CVSS Exploitability score as the only variable in the set $\mathcal{X}_l^{\uparrow}$ defined by Equation 10 and the CVSS Impact score as the only variable in the set $\mathcal{X}_e^{\uparrow}$ defined by Equation 10. No variables were considered for the other two sets, $\mathcal{X}_l^{\downarrow}$ and $\mathcal{X}_e^{\downarrow}$. Equation 14 and 17 can be rewritten as follows for scenario 3.

$$\rho(v) = \prod_{X \in \{\text{CVSS\_Exploitability}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

$$ef(v) = \prod_{X \in \{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 6, the quality of the resulting ranking starts to improve as the combined effect of multiple variables allows to better discriminate between different vulnerabilities, although the ranking is still far from the ideal case. The top 10 values of the severity score now involve "only" 2430 CVEs, an order of magnitude less than the previous 2 scenarios.

This scenario is directly comparable to a scenario in which CVSS Base Score is used to rank vulnerabilities. Although our severity score formula is different from the CVSS Base Score formula, both are ultimately functions of two variables that can assume only a limited number of discrete values, 27 and 10 respectively, thus limiting their ability to provide a useful ranking of vulnerabilities. With subsequent scenarios, we show how considering more variables improves the quality of the ranking.

6.4. Scenario 4

In this scenario, in addition to the CVSS Exploitability score and the CVSS Impact score, we considered the set of known IDS rules as a variable in $\mathcal{X}_l^{\downarrow}$, with f_X defined as the cardinality of the set of rules. Equation 14 and 17 can be rewritten similarly to what we have shown for previous scenarios.

Table 7
Ranking of CVEs in Scenario 4.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.50156	821	821	820.0000	0.03%
2	0.49163	35	856	580.3258	0.30%
3	0.48115	10	866	473.8625	0.88%
4	0.47695	712	1578	542.9452	0.44%
5	0.46867	10	1588	485.6416	0.78%
6	0.46751	7	1595	443.3349	1.19%
7	0.46532	730	2325	494.3561	0.71%
8	0.45978	2	2327	462.4280	0.98%
9	0.45755	11	2338	435.9940	1.28%
10	0.45611	26	2364	413.6958	1.60%

Table 8
Ranking of CVEs in Scenario 5.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.33152	1	1	0.0000	100.00%
2	0.31910	1	2	0.0000	100.00%
3	0.31229	3	5	1.1547	98.85%
4	0.29519	2	7	1.1180	98.89%
5	0.28825	8	15	3.2863	96.77%
6	0.27745	2	17	3.0277	97.02%
7	0.27153	4	21	3.0237	97.02%
8	0.25819	1	22	2.8284	97.21%
9	0.24028	2	24	2.6874	97.35%
10	0.19558	1	25	2.5495	97.48%

$$\rho(v) = \frac{\prod_{X \in \{\text{CVSS\_Exploitability}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X \in \{\text{Known\_IDS\_Rules}\}} e^{\beta_X \cdot f_X(X(v))}}$$

$$ef(v) = \prod_{X \in \{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 7, the quality of the resulting ranking improves slightly compared to scenario 3. The relatively small increase is due to the fact that most CVEs have either no or only one associated IDS rule. However, new rules can be defined over time.

6.5. Scenario 5

In this scenario, in addition to the variables considered in Scenario 4, we considered the set of vulnerability exploits potentially available to the attacker as a variable in $\mathcal{X}_l^{\uparrow}$, with f_X defined as the cardinality of the set of exploit. Equation 14 and 17 can be rewritten similarly to what we have shown for previous scenarios.

$$\rho(v) = \frac{\prod_{X \in \{\text{CVSS\_Exploitability, Vuln\_Exploit}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X \in \{\text{Known\_IDS\_Rules}\}} e^{\beta_X \cdot f_X(X(v))}}$$

$$ef(v) = \prod_{X \in \{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 8, the quality score of the ranking improves significantly, reaching 97% for $r = 10$. For $r = 2$ the quality score is 100% as there is exactly one CVE for each of the top 2 severity scores. This can be explained by considering that vulnerability exploits are publicly available for a relatively small number of vulnerabilities, thus favoring these vulnerabilities over many others. It is also worth nothing that, as more variables are considered, the highest severity score becomes smaller. This is expected, as we are multiplying more factors between 0 and 1, but this does not affect our results, as we are interested in the relative ranking of vulnerabilities rather than in their absolute scores.

Table 9
Ranking of CVEs in Scenario 6.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.33152	1	1	0.0000	100.00%
2	0.31910	1	2	0.0000	100.00%
3	0.31229	2	4	0.5774	99.42%
4	0.29519	2	6	0.7071	99.30%
5	0.28825	8	14	3.1937	96.86%
6	0.27745	2	16	2.9439	97.10%
7	0.27153	4	20	2.9520	97.09%
8	0.25819	1	21	2.7613	97.28%
9	0.24321	1	22	2.6034	97.43%
10	0.24028	2	24	2.4900	97.54%

Table 10
Ranking of CVEs in Scenario 7.

Rank	Score	# CVEs	Cumulative	Deviation	Quality Score
1	0.84228	1	1	0.0000	100.00%
2	0.84227	1	2	0.0000	100.00%
3	0.84226	4	6	1.7321	98.28%
4	0.84226	1	7	1.5000	98.51%
5	0.84226	1	8	1.3416	98.67%
6	0.84226	3	11	1.4720	98.54%
7	0.84225	3	14	1.5584	98.45%
8	0.84225	1	15	1.4577	98.55%
9	0.84224	1	16	1.3744	98.64%
10	0.84224	2	18	1.3416	98.67%

6.6. Scenario 6

In this scenario, in addition to the variables considered in Scenario 5, we considered the set of vulnerability scanning plugins as a variable in $\mathcal{X}_l^{\downarrow}$, with f_X defined as the cardinality of the set of plugins. Eqs. 14 and 17 can be rewritten similarly to what we have shown for previous scenarios.

$$\rho(v) = \frac{\prod_{X\in\{\text{CVSS\_Exploitability,Vuln\_Exploit}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X\in\{\text{Known\_IDS\_Rules,Vuln\_Plugins}\}} e^{\beta_X \cdot f_X(X(v))}}$$

$$ef(v) = \prod_{X\in\{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 9, the quality score of the ranking improves slightly compared to Scenario 5. The almost negligible improvement may be explained with these two arguments: (i) once the quality score has reached over 95%, additional improvements cannot be expected to be significant due to a diminishing returns effect; and (ii) the vulnerabilities for which the hacker community has developed exploits may be those that vulnerability scanning vendors prioritize in the development of plugins. In other words, if two variables are highly correlated, including both of them in the computation of vulnerability metrics may produce only a slight improvement over scenarios in which only one of them is used.

6.7. Scenario 7

In this scenario, we consider the variables considered in Scenario 4 and the age of a vulnerability. Equation 14 and 17 can be rewritten as follows.

$$\rho(v) = \frac{\prod_{X\in\{\text{CVSS\_Exploitability,Age}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)}{\prod_{X\in\{\text{Known\_IDS\_Rules}\}} e^{\beta_X \cdot f_X(X(v))}}$$

$$ef(v) = \prod_{X\in\{\text{CVSS\_Impact}\}} \left(1 - e^{-\alpha_X \cdot f_X(X(v))}\right)$$

As shown in Table 7, the quality score of the resulting ranking is comparable to that obtained in the previous two scenarios. Once again, this can be explained by considering that the age of a vulnerability is correlated with the availability of both exploits and plugins. In fact, as the vulnerability ages, more plugins and exploits are developed.

7. Conclusions

In this paper, to address the lack of principled approaches to quantify various dimensions of vulnerabilities and provide scoring capabilities that can adapt to different domains, we introduced a vulnerability metrics framework that extends and generalizes our previous metrics for evaluating the exploitation likelihood of a vulnerability and the exposure factor of system components to vulnerability exploits. We showed how the choice of variables to consider in the calculation of either metric can influence the quality of the resulting ranking. As part of our future work, we plan to further investigate the idea that attacker and defender models can inform the selection of variables to consider, and we will also study the correlation between different variables in order to avoid unnecessary calculation that do not help refine the ranking.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Massimiliano Albanese: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Ibifubara Iganibo:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Software, Writing – original draft. **Olutola Adebiyi:** Methodology, Validation, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing.

Acknowledgements

This work was funded in part by the National Science Foundation under award CNS-1822094.

References

Albanese, M., Jajodia, S., 2018. A graphical model to assess the impact of multi-step attacks. Journal of Defense Modeling and Simulation 15 (1), 79–93. doi:10.1177/1548512917706043.

Albanese, M., Pugliese, A., Subrahmanian, V., 2013. Fast activity detection: indexing for temporal stochastic automaton-based activity models. IEEE Trans Knowl Data Eng 25 (2), 360–373. doi:10.1109/TKDE.2011.246.

Ammann, P., Wijesekera, D., Kaushik, S., 2002. Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002). ACM, Washington, DC, USA, pp. 217–224. doi:10.1145/586110.586140.

Bahl, P., Barham, P., Black, R., Chandra, R., Goldszmidt, M., Isaacs, R., Kandula, S., Li, L., MacCormick, J., Maltz, D., Mortier, R., Wawrzoniak, M., Zhang, M., 2006. Discovering dependencies for network management. In: Proceedings of the 5th ACM Workshop on Hot Topics in Networking (HotNets-V). ACM, Irvine, CA, USA, pp. 97–102.

Bopche, G.S., Rai, G.N., Denslin Brabin, D.R., Mehtre, B.M., 2019. A proximity-based measure for quantifying the risk of vulnerabilities. In: Thampi, S.M., Perez, G.M., Ko, R., Rawat, D.B. (Eds.), Proceedings of the 7th International Symposium on Security in Computing and Communication (SSCC 2019). Springer, pp. 41–59. doi:10.1007/978-981-15-4825-3_4.

Gallon, L., 2010. On the impact of environmental metrics on CVSS scores. In: Proceedings of the 2010 IEEE International Conference on Social Computing (SocialCom 2010). IEEE, Minneapolis, MN, USA, pp. 987–992. doi:10.1109/SocialCom.2010.146.

Goohs, J., Mier, R., Deist, P., Casey, W., 2022. Reducing attack surface by learning adversarial bag of tricks. In: Proceedings of the 21st Workshop on the Economics of Information Security (WEIS 2022). Tulsa, OK, USA.

Iganibo, I., Albanese, M., Mosko, M., Bier, E., Brito, A.E., 2021. Vulnerability metrics for graph-based configuration security. In: Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021). SciTePress, pp. 259–270. doi:10.5220/0010559402590270.

Iganibo, I., Albanese, M., Turkmen, K., Campbell, T., Mosko, M., 2022. Mason Vulnerability Scoring Framework: a customizable framework for scoring common vulnerabilities and weaknesses. In: Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022). SciTePress, Lisbon, Portugal, pp. 215–225. doi:10.5220/0011277400003283.

Jacobs, J., Romanosky, S., Edwards, B., Adjerid, I., Roytman, M., 2021. Exploit Prediction Scoring System (EPSS). Digital Threats: Research and Practice 2 (3). doi:10.1145/3436242.

Jajodia, S., Noel, S., O'Berry, B., 2005. Managing Cyber Threats: Issues, Approaches, and Challenges. In: Massive Computing, Vol. 5. Springer, pp. 247–266.

Kourtellis, N., De Francisci Morales, G., Bonchi, F., 2015. Scalable online betweenness centrality in evolving graphs. IEEE Trans Knowl Data Eng 27 (9), 2494–2506. doi:10.1109/TKDE.2015.2419666.

Leversage, D.J., Byres, E.J., 2008. Estimating a system's mean time-to-compromise. IEEE Security & Privacy 6 (1), 52–60. doi:10.1109/MSP.2008.9.

Manadhata, P.K., Wing, J.M., 2011. An attack surface metric. IEEE Trans. Software Eng. 37 (3), 371–386. doi:10.1109/TSE.2010.60.

Natrajan, A., Ning, P., Liu, Y., Jajodia, S., Hutchinson, S.E., 2012. NSDMiner: Automated discovery of network service dependencies. In: Proceedings of the 31st Annual IEEE International Conference on Computer Communications (IEEE INFOCOM 2012). IEEE, Orlando, FL, USA, pp. 2507–2515. doi:10.1109/INFCOM.2012.6195642.

Soroush, H., Albanese, M., Asgari Mehrabadi, M., Iganibo, I., Mosko, M., Gao, J.H., Fritz, D.J., Rane, S., Bier, E., 2020. SCIBORG: Secure configurations for the IoT based on optimization and reasoning on graphs. In: Proceedings of the 8th IEEE Conference on Communications and Network Security (CNS 2020). IEEE doi:10.1109/CNS48642.2020.9162256.

Spring, J., Hatleback, E., Householder, A., Manion, A., Shick, D., 2021. Time to change the CVSS? IEEE Security & Privacy 19 (2), 74–78. doi:10.1109/MSEC.2020.3044475.

Stuckman, J., Purtilo, J., 2012. Comparing and applying attack surface metrics. In: Proceedings of the 4th International Workshop on Security Measurements and Metrics (MetriSec 2012). ACM, Lund, Sweden, pp. 3–6. doi:10.1145/2372225.2372229.

van der Stock, A., Glas, B., Smithline, N., Gigler, T., 2017. OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks. Technical Report. The OWASP Foundation.

van der Stock, A., Glas, B., Smithline, N., Gigler, T., 2021. OWASP Top 10 for 2021: The Ten Most Critical Web Application Security Risks. Technical Report. The OWASP Foundation.

Venkatesan, S., Albanese, M., Jajodia, S., 2015. Disrupting stealthy botnets through strategic placement of detectors. In: Proceedings of the 3rd IEEE Conference on Communications and Network Security (IEEE CNS 2015). IEEE, Florence, Italy, pp. 55–63. doi:10.1109/CNS.2015.7346816.

Yoon, S., Cho, J.-H., Kim, D.S., Moore, T.J., Free-Nelson, F., Lim, H., 2020. Attack graph-based moving target defense in software-defined networks. IEEE Trans. Netw. Serv. Manage. 17 (3), 1653–1668. doi:10.1109/TNSM.2020.2987085.