




Article

Automated Context-Aware Vulnerability Risk Management for Patch Prioritization

Vida Ahmadi Mehri ^{1,*} , Patrik Arlos ¹  and Emiliano Casalicchio ^{1,2} ¹ Department of Computer Science, Blekinge Institute of Technology, 37179 Karlskrona, Sweden² Department of Computer Science, Sapienza University of Rome, 00168 Rome, Italy

* Correspondence: vida.ahmadi.mehri@bth.se (V.A.M.)

Abstract: The information-security landscape continuously evolves by discovering new vulnerabilities daily and sophisticated exploit tools. Vulnerability risk management (VRM) is the most crucial cyber defense to eliminate attack surfaces in IT environments. VRM is a cyclical practice of identifying, classifying, evaluating, and remediating vulnerabilities. The evaluation stage of VRM is neither automated nor cost-effective, as it demands great manual administrative efforts to prioritize the patch. Therefore, there is an urgent need to improve the VRM procedure by automating the entire VRM cycle in the context of a given organization. The authors propose automated context-aware VRM (ACVRM), to address the above challenges. This study defines the criteria to consider in the evaluation stage of ACVRM to prioritize the patching. Moreover, patch prioritization is customized in an organization's context by allowing the organization to select the vulnerability management mode and weigh the selected criteria. Specifically, this study considers four vulnerability evaluation cases: (i) evaluation criteria are weighted homogeneously; (ii) attack complexity and availability are not considered important criteria; (iii) the security score is the only important criteria considered; and (iv) criteria are weighted based on the organization's risk appetite. The result verifies the proposed solution's efficiency compared with the Rudder vulnerability management tool (CVE-plugin). While Rudder produces a ranking independent from the scenario, ACVRM can sort vulnerabilities according to the organization's criteria and context. Moreover, while Rudder randomly sorts vulnerabilities with the same patch score, ACVRM sorts them according to their age, giving a higher security score to older publicly known vulnerabilities.

Keywords: vulnerability management; risk management; security management; patch prioritization

Citation: Ahmadi Mehri, V.; Arlos, P.; Casalicchio, E. Automated Context-Aware Vulnerability Risk Management for Patch Prioritization. *Electronics* **2022**, *11*, 3580. <https://doi.org/10.3390/electronics11213580>

Academic Editors: Jheng-Jia Huang, Ray-Lin Tso and Po-Wen Chi

Received: 6 September 2022

Accepted: 24 October 2022

Published: 2 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Vulnerability risk management (VRM) is one of the critical aspects of information security. Many of today's cyberattacks exploit systems' vulnerabilities (e.g., CVE-2021-40444, CVE-2021-44228, CVE-2021-3156) [1]. Unpatched vulnerabilities caused 60% of known data breaches, according to [2]. Hence, VRM is a fundamental part of information-security management in every organization. It consists of identifying, classifying, evaluating, and remediating vulnerabilities.

According to [3], the identification of vulnerabilities by vulnerability scanner tools (such as OpenVAS [4] and Nessus [5]) is only a small part of the VRM process. Security experts use these scanners to inspect their systems regularly. In addition, security experts' knowledge of an organization is crucial to evaluating the risk of exploits and prioritizing the order of patches. The evaluation and prioritizing of remediation are the challenging parts of the VRM process. The requirement to involve security experts and the dramatic increase in known vulnerabilities in the last five years (+26%) (2016–2021) have made the VRM process time-consuming and expensive. The research question is, therefore: how to make the VRM process time efficient, cost effective, and organization oriented?

To answer the above research question, we introduced the concept of automated context-aware vulnerability risk management (ACVRM) [6,7]. ACVRM facilitates the customization of the VRM process for a given organization by learning about the organization's assets and the vulnerabilities that affect these assets. ACVRM automates the VRM process by applying predefined decision criteria and related activities, thus saving time and cost.

In our previous studies [6,7], we identified that the selection of what vulnerability database (VD) to use plays an essential role in the VRM procedure and the information on an organization's assets should support VD choice. Indeed, the vulnerability severity score comes from a VD, and there are several types of VD, from national [8] to vendor [9,10], and even application-specific [11]. Vendors' VD, such as RedHat [9], usually list their affected releases, severity score in their environments, and patch instructions for the vulnerabilities that affect their products. In contrast, national VD's provide general information about the vulnerability and a severity score. Unfortunately, existing vulnerability scanner tools do not allow the selection of the VD to use. Moreover, all of them rely on a single VD. In addition, scanners do not know the system architecture and an organization's configuration policy to identify the actual exposure of the vulnerability in the organization. Therefore, security experts should define criteria in the evaluation step in VRM to prioritize the remediation of the vulnerabilities. This research enhances our previous work [6,7] focusing on the VD-selection problem and on the challenge of defining evaluation criteria for context-aware patch prioritization. Our contribution is summarized as follows:

- We present the prioritization-phase workflow of the ACVRM framework, describing the details of the filter, evaluation, and patch prioritization stages.
- We define a *patch-score* criteria to prioritize patching which could be adapted to an organization's context. The criteria build on security experts' interviews and a literature study.
- We implement a proof of concept (PoC) of the ACVRM framework.
- We validate the ACVRM PoC against the prioritization obtained using the Rudder tool. In the evaluation, we consider four case studies for the organizational context that impact the Patch Score. Results show the capability of ACVRM in customizing patch prioritization.

Figure 1 shows the relationship between our previous work and the current work. The initial idea for ACVRM was presented in [6], and in that study, the focus was on calculating a normalized vulnerability score based on multiple vulnerabilities. This work was then augmented in [7], where we investigated the impact that the selection of VD or VDs has on the obtained score. These two works are centered around the first phase, retrieval and pre-processing in the ACVRM, see Figure 2. This work focuses on the second phase, prioritization. The third phase, patch management is left out for the moment, as most organizations use tools to apply patches.

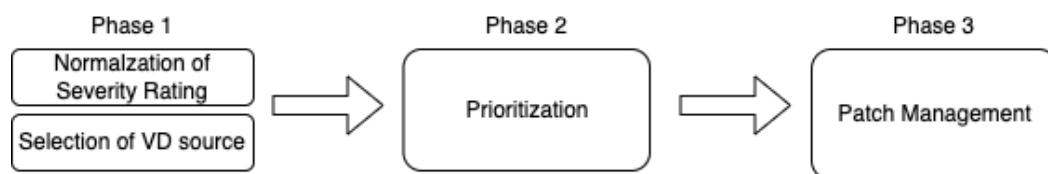


Figure 1. Relationship between our current work (phase 2) and previous work (phase 1), and how they address the ACVRM phases.

The paper is organized as follows. Section 2 provides background on VRM and analyzes the related literature. Section 3 introduces the ACVRM framework, and Section 4 describes the ACVRM's prioritization phase, where the paper's core contribution is. Section 5 presents the selection of the evaluation criteria and the definition of the patch score. Section 5 describes the design and implementation of a proof of concept for the prioritization phase. Experiments and results are reported in Sections 7 and 8, respectively. Section 9 concludes the paper.

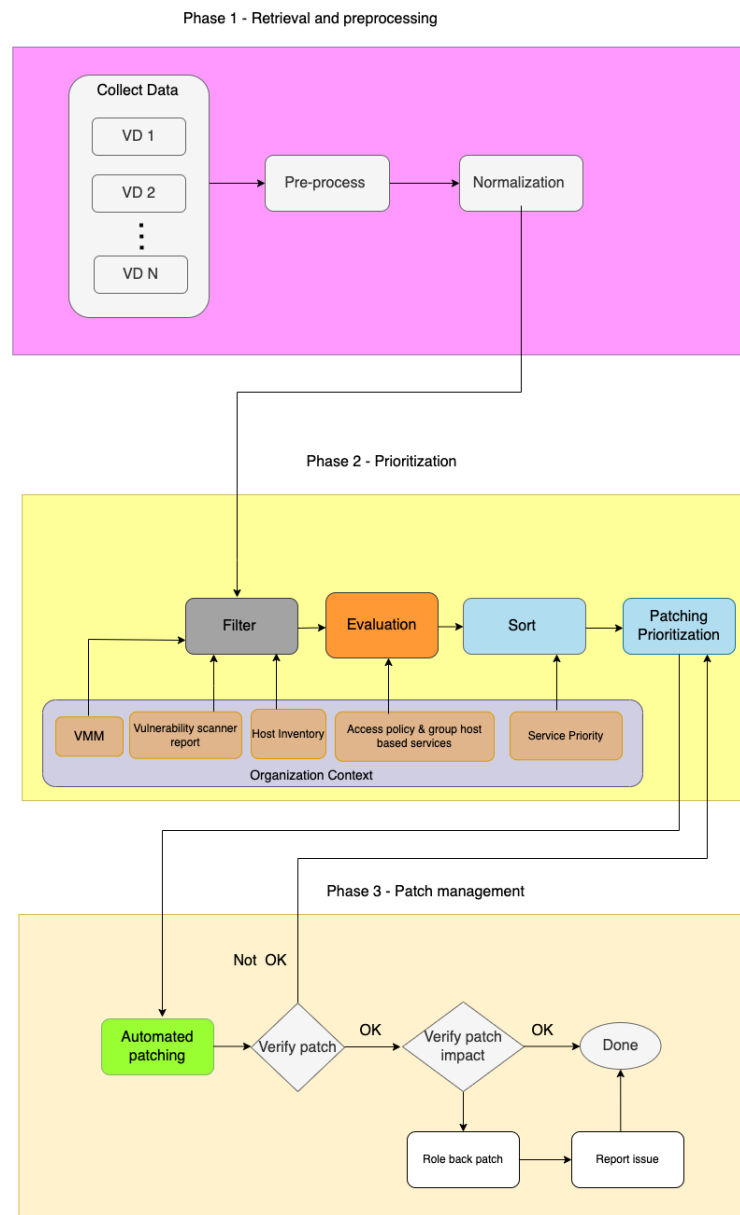


Figure 2. The ACVRM phases.

2. Related Work

Continuous VRM is in the top-10 critical security controls defined by the Center of Internet Security (CIS) [12]. VRM is one of the vital criteria to guarantee system compliance. Most information security standards (e.g., ISO 27002, PCI, SOC2) and legislation (e.g., EU Cybersecurity act [13], EU Cybersecurity Certificate (EUCS) [14], USA homeland security act [15]) include VRM as a critical control. Hence, organizations must establish a VRM process to remediate the identified vulnerabilities.

Keeping up with assessing hundreds of vulnerabilities daily is a big challenge for the security teams in every organization. It is impossible to patch all detected vulnerabilities due to resources and time limitations. Therefore, most tools and security analysts prioritize remediation based on severity score. The severity score can be calculated using the Common Vulnerability Scoring System (CVSS). CVSS is an open framework which transfers the vulnerability characteristic to a numeric score [16]. The score obtained from CVSS is static, and the numeric value of each metric does not change over time. To overcome this problem, researchers proposed a methodology to change the numeric value of impact

metrics (i.e., confidentiality, integrity, and availability (CIA)) in the CVSS version 2.0 in favor of improving the CVSS scoring [17]. The authors found that the violation of confidentiality is more severe than integrity and availability, and, hence, should not be weighted equally. Another approach proposed to improve patch prioritization in the VRM process is to add temporal and environmental metrics to the CVSS score [18]. Rather than adding new metrics to the CVSS or changing CVSS information over time, we propose to feed the VRM process with the Organization Context (OC) data. Indeed, in ACVRM, the organization context data complements the CVSS information in evaluating the vulnerability ranking. The OC is the set of data that defines the assets the organization intends to protect, and the rules. The OC data we propose to use in ACVRM are described in Sections 4.1 and 4.2.

According to the Ponemon Institute [2], 32% of a survey's participants made a remediation decision based on the CVSS score. A total of 59% of participants in the survey disclosed that their organizations were not performing the complete VRM's life cycle. A gap in the VRM life cycle is seen as an opportunity for adversaries to leverage vulnerabilities. The 2021 Check Point Cyber Security Report [19] reveals that 80% of attacks in 2020 took advantage of vulnerabilities reported in 2017 or earlier. Furthermore, around 50% of the participants in the Ponemon survey [2] recognized that automation is a key to responding to a vulnerability promptly. To address the above-mentioned issue, we designed ACVRM to adapt its behavior based on the organization context to prioritize remediation.

Many studies have applied machine-learning-based solutions to predict remediation decisions and classify the type of vulnerability in different domains such as power grid and software development. For example, authors in [20] built their decision tree based on data of the asset and vulnerability features for a power grid and reached 97% accuracy. However, their solution is domain-specific and requires manual verification on the small prediction portion to reduce false negatives. On the contrary, in ACVRM, we propose to improve patch prioritization over time based on the historical organizational data from the feedback loop.

Vulnerability categorization is also helpful in automating VRM and in the software development life cycle. In [21], the authors propose using multiple machine-learning algorithms to classify vulnerabilities into vulnerability categories, as understanding vulnerability types is crucial in the software development life cycle. Similarly, a machine-learning algorithm allows the classification of vulnerability types in a security patching of open-source software [22]. In our solution, we foresee to apply, as future work, a machine-learning algorithm to improve patch prioritization based on the patch verification feedback.

In [23], the authors proposed the automated CVSS-based vulnerability prioritization. Their solution uses only the vulnerability scanner report of the environment and prioritizes the patch based on the confidentiality, integrity, and availability score. The authors concluded that using a CVSS-based score is insufficient, and they should consider other metrics in a prioritization step in the future. SmartPatch is a patch-prioritization method for Industrial Control Systems [24]. SmartPatch uses the network topology of environments and the vulnerability scanner report to address patch sequencing in an interdependent and complex network. SmartPatch proposed a security metric called Residual Impact Score (RIS) by utilizing the score of the impact metrics and exploitability metrics of CVSS exported from the National Vulnerability Database (NVD). The authors in [25] used a mathematical approach to select the vulnerability from the scan report for remediation concerning the available experts. They used the CVSS score from NVD, the available hours of security experts, the vulnerability's age, and its persistence in the monthly scan in their approach. They concluded that the number of unpatched vulnerabilities was the lowest using multiple attributes. In [26], the authors proposed a machine-learning-based method to address the inconsistency of CVSS scores across multiple VDs. They trained their algorithm with a different version of CVSS scores in NVD and validated their result with crawled vulnerability reports from SecurityFocus. Then, they implemented the case study in cyber-physical systems to assess the severity of the vulnerability. The result of their case study indicated the diversity of vulnerability scores on different data sources which mislead the experts in patch prioritization.

Compared to the above research work ([23–26]), ACVRM facilitates patch prioritization for organizations independently of the domain. It utilized multiple VDs, host inventory, and scan reports to detect existing vulnerabilities. ACVRM also customized the VRM procedure for the organization by enabling them to select the vulnerability management mode (VMM) and to weigh the criteria used in patch prioritization.

Table 1 summarizes the comparison between our solution and the most recent state-of-the-art works on vulnerability prioritization. The comparison is performed according to the following features: the VD used as reference (Reference VD), the vulnerability identification approach, the vulnerability evaluation criteria, and the contribution provided. The comparison highlights the following: our solution is the only one that allows multiple VDs as input for vulnerability identification; there is a shared consensus on using multiple sources of information to identify vulnerabilities and using multiple evaluation criteria. Concerning evaluation criteria, while the majority of the proposed solutions use the security score (SC), confidentiality (C), integrity (I), and availability (A), ACVRM also adopts the attack vector (AV) and attack complexity (AC) along with the access level (internal or external AUS) metrics. The complexity of attacks is also addressed by other works using the exploitation rate (ER) or collateral damage potential (CDP) metrics.

Table 1. Comparison of our approach to most recent related work. For acronyms, cf. Section 5 and the Abbreviations section.

Reference VD	Identification Approach	Evaluation Criteria	Contribution
Walkowski et al. [23] NVD	Vulnerability scan	SC, C, I, A, CDP	Proposed VRM improvement by prioritizing the patch based on the CDP value for monitored IT sources and the ratio of detected vulnerabilities to the number of monitored resources
Yadav et al. [24] NVD	Vulnerability scan and network topology	SC, C, I, A, ER, functional and topological dependencies	Defined security metric Residual Impact Score (RIS) used to prioritise the patch based on cost of defence, cost of attack, and impact of attack
Jiang et al. [26] NVD and security-Focus	Vulnerability scan and system configuration	SC	Proposed machine-learning-based structure to correlate SC from multiple sources to overcome inconsistency in CVSS score
Shah et al. [25] NVD	Vulnerability scan	SC, age, and persistence	Defined mathematical model for optimizing remediation priority with respect to evaluation criteria
Our work NVD, DSA, RHSA, and USN	Vulnerability scan, asset inventory, and VDs data	SC, C, I, A, AV, AC, and Access level	Proposed the criteria for patch prioritization and customised the patch in the organization's context

3. Automated Context-Aware Vulnerability Risk Management (ACVRM)

ACVRM aims to improve the VRM as follows: (1) it uses multiple VDs for retrieving common vulnerabilities and exposures (CVE) [27] data; (2) it automates the classification of vulnerabilities and the patch-prioritization process based on an organization's requirements. ACVRM is structured into three phases, as shown in Figure 2. Phase 1 has been addressed in our previous works [6,7], phase 2 design and implementation is the main contribution of this paper; and phase 3 is considered to be future work.

During phase 1, ACVRM retrieves CVE data from multiple VDs. From each VD, we collect CVE-IDs, their publication date, description, severity score, affected releases, and

safe version. We also store a timestamp to know when we collected or updated the data in our local database. To keep the local database updated, ACVRM periodically checks for changes in the source VD. If changes are detected, the local database is updated while keeping the old version of CVE-ID data in an archive for future reference. The pre-process stage converts the quantitative severity score of CVE-ID (if any) to an internal numeric score using the conversion algorithm described in [7]. Our internal score is based on the CVSS 3.x score. The pre-process stage makes the CVE-ID data ready for the normalization stage. The main task in the normalization stage is calculating a severity score for each CVE-ID. ACVRM offers three VMM: basic, standard, and restrictive. These are in-line with the three assurance levels proposed by EUCS[14]. A basic VMM is the minimum acceptable baseline for a VRM process suggested for an organization with a limited risk of exploitation (e.g., an organization with a limited system exposed to the Internet). Standard VMM is suitable to serve an organization with medium-to-high-security risks. At the same time, the restrictive VMM should be used in compliant organizations (i.e., an organization or governmental agency that should comply with local and international regulations or standards and critical infrastructure). The normalization stage calculates the normalized score for each CVE-ID by averaging the severity scores concerning VMM mode.

In phase 2, ACVRM determines the patch prioritization for an organization’s needs, specified by the organization’s context. Phase 2 is the core part of ACVRM and is described in detail in Section 4.

In the third phase, patch management, ACVRM patches the detected vulnerabilities and verifies that the system functionalities are not compromised.

In the first stage of patch management, the automated patching component executes patch prioritization on vulnerable hosts. Then, ACVRM verifies if the patch was successful. If an error occurs for an item in the patch-prioritization list, it will jump to the next item in the list and record the one encountered in an error state. If an error appears due to the patch order (e.g., patching a microcode vulnerability in Ubuntu requires the kernel to be patched in advance), it will re-execute the patch at the end. For persistent errors, the report with the error state will submit to the patch-prioritization stage in phase 2 for review. The final stage in phase 3, verification, consists of evaluating the impact of the patch on the application/unit/service’s (AUS) functionality. The functional tests refer to the series of predefined tests by experts to investigate the health of AUS. In case of unexpected behavior, the issue is reported to the experts.

4. Prioritization

Prioritization is the second phase of ACVRM, which determines the order of CVE-IDs to be patched in each host. Figure 3 shows the stages in the prioritization phase. In the following sections, we will briefly describe these stages.

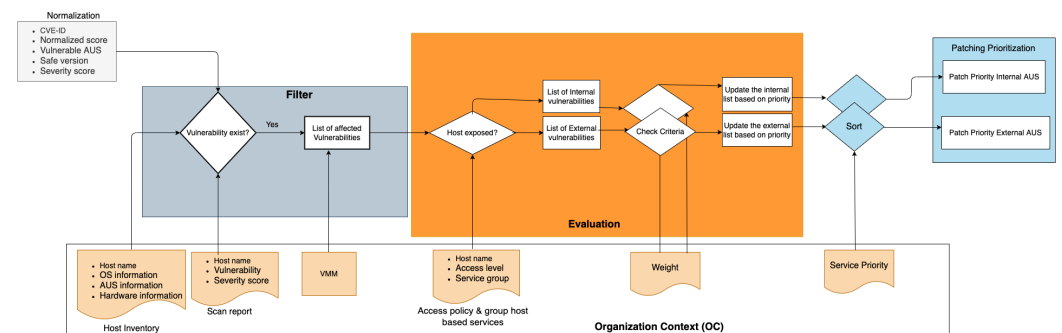


Figure 3. The phase 2—prioritization process.

4.1. Filter

The task of the filter stage is to identify vulnerabilities that affect the organization’s assets (i.e., application, software, servers). The inputs are the data from the normalization

stage, the host inventory, and the vulnerability scan reports provided with the OC, cf. Figure 3. The output is the list of vulnerabilities affecting the organization's assets. The list includes CVE-IDs, hostname, name of vulnerable AUS, and normalized score. In more detail, the OC data used in the filter stage are:

- Host inventory (HI): consisting of the hosts and assets belonging to the organization. The host inventory provides the list of hosts, hardware specifications, and installed software. Examples are asset-management tools (e.g., Device42, NinjaRRM, Solarwinds) or custom tools.
- Vulnerability scanner report (VSR): a source used by security specialists for patch prioritization. The best security practices for the cloud, such as C5 [28], suggest monthly vulnerability scanning, which leads to thirty days of patch planning.
- Vulnerability management mode (VMM): defines in which mode ACVRM should operate. The organization sets a default VMM mode for the whole organization, but this can be overridden with host-based VMM, e.g., VMM basic for host A, VMM restricted for host B, while the default is standard.

The filter identifies the CVE-IDs that impact the organization by comparing the vulnerable software and their existence in the *HI* and *VSR*. For example, a vulnerability scanner might report faulty configurations with no CVE-ID reference (i.e., Nessus ID 153953: SSH server configured to allow weak key exchange algorithms). We might also find vulnerable software installed in the hosts in the organization's environment but not detected by the vulnerability scanner (i.e., the vulnerability in sudo before 1.9.5p2 (CVE-2021-3156) that the Nessus scanner did not discover in our test environment). Therefore, we obtain better coverage of the potential vulnerabilities by considering both the *HI* and *VSR*. Let *C* represent all collected vulnerabilities; then, the filter will produce a list of vulnerabilities (*VULN*) that are affected by the *VSR* and/or *HI*

$$VULN = (C \cap C_{VSR}) \cup \Lambda(C, HI)$$

where C_{VSR} is the set of vulnerabilities contained in the *VSR*; and $\Lambda(C, HI)$ is a filter function that returns only the vulnerabilities that affect the hosts.

4.2. Evaluation

The task in the evaluation stage is to examine the risk of each vulnerability and provide the patch prioritization. The input for the evaluation stage is the list of affected vulnerabilities from the filter stage, the access policy and group host-based services, and the weight from the OC.

The access policy and group-based services describe conceptual information to enforce business requirements. It defines access to applications/services based on the host group, locations, and time. The access policy provides information on how accessible different AUS are, i.e., AUS exposed to the public are probably more likely to be compromised than AUS that are not. The group-based services simplify the patch process as the same patching and verification instructions will apply.

In the evaluation stage, we divide the list of affected vulnerabilities based on the access policy into external and internal groups. The external refers to the AUS being exposed to the Internet, and, thus, having a higher risk of exploitation. On the other hand, internal indicates the AUS with limited access levels, i.e., authorized users with defined IP addresses in the access control list (ACL).

In this stage, we also check the criteria that impacted the patch sequence. The organization could customize the criteria by weighting them based on the impact on its business, cf. Section 5.

4.3. Sort

The task for the sort stage is to update the order of the CVE in each evaluated list (i.e., external list). The inputs are the output from the evaluation stage and the service policy.

The service policy is optional information to influence the order of the patch list for the vital services for the organization. The services listed in service priority are granted a higher position in the patch list. If the organization does not provide service priority, the sort will be based on the PS score, cf. Section 5.4. The outputs are two sorted lists of vulnerabilities for internal and external AUS. Each list includes CVE-IDs, hostname, name of vulnerable AUS, PS, and priority number.

4.4. Patching Prioritization

The main task for patching prioritization is to adjust the patch order based on learning. This stage receives the error feedback from patch verification. It builds the knowledge to map situations to actions over time. The patch prioritization input is the sort stage output and the feedback loop from the phase 3 of ACVRM. In the first round, the patch-prioritization stage provides the same output as the sort stage, as it is not yet received any feedback from phase 3.

5. Evaluation Criteria and Patch Score

Finding a suitable criteria for evaluating vulnerabilities is a challenge, as demonstrated by the multiple research studies on this subject [18,23,24,29–38]. As we mentioned earlier in Section 1, the evaluation stage depends on expert and organizational knowledge. Automating the vulnerability evaluation procedure is crucial for each organization because some vulnerabilities might remain unpatched in a system due to there being many vulnerabilities and a limited number of available security experts. We applied the following methods to define evaluation criteria and automate the evaluation stage to address the challenges mentioned earlier:

1. We reviewed the scientific papers on vulnerability patch prioritization to find evaluation criteria for ranking the vulnerabilities.
2. We interviewed security experts with different seniority levels in VRM to manually rank the criteria they are using to prioritize the vulnerability patch.
3. We analyzed the obtained criteria from items 1 and 2 to introduce a patch score(PS). PS is a mathematical approach to calculating the priority of each vulnerability from the evaluation criteria and their weight based on the organizational context.

In this section, we described our methods of finding the criteria in detail and how we can customize the PS in the organization's context.

5.1. Analysis of Vulnerability Evaluation Criteria in the Literature

We conducted our search in Google Scholar because it is a comprehensive academic search engine with 389 million records [39]. The selected search string “vulnerability patch priority” was applied to identify the patch-prioritization criteria in the relevant literature. The search query indicates that the string should include the title and abstract of a peer-reviewed publication. Then, we excluded the papers that were not relevant to the goal of this paper based on the title and abstract. Finally, we performed a full-text assessment of fifteen selected papers.

From these fifteen papers, we identified nine criteria, reported in Table 2: a ✓ sign means the criterion is considered in the paper. The related work review shows that the severity score is a common criterion. In addition, ten of fifteen (66.7%) studies recognize the CVSS impact metrics, confidentiality, integrity, and availability as critical metrics in priority decisions. We also observed that eleven of fifteen (73.3%) papers identify the exploitation rate (similar to attack complexity in CVSS v3) as an essential criterion. The considered criteria in the related papers are described as follows:

- Severity score (SC) is a transferring of the vulnerability characteristics to a numeric score between 0 to 10.
- Confidentiality (C) measures the impact of the disclosure of the information to an unauthorized party due to a successfully exploited vulnerability.

- Integrity (I) refers to the impact of altering information by an unauthorized user on the trustworthiness of data due to a successfully exploited vulnerability.
- Availability (A) measures the impact of a successfully exploited vulnerability on the system and data accessibility.
- Age/time is a time difference between the CVE-ID published date and the current date.
- Common configuration enumeration (CCE) [40] is a unique identifier for system configuration issues and provides accurate configuration data across multiple tools and sources of information. CCE serves as a configuration best practice.
- Collateral damage potential (CDP) is an environmental metric in CVSS V2 and refers to loss of life, physical assets, productivity, or revenue. Modified base metrics replaced CDP in CVSS V3 to reduce the impact of successfully exploiting the vulnerability by enforcing a change in the default configuration of a vulnerable component.
- Exploitation rate (ER) provides the rate of how likely the vulnerability is to be exploited. CVSS V3 addresses ER in the attack complexity (AC) metric, which evaluates the amount of effort required to exploit the vulnerable component.
- Vulnerability type (VT) refers to the attacker's activity as a result of successfully exploiting vulnerabilities such as denial of services (DoS), code execution, privilege escalation, and buffer overflow.

Table 2. The evaluation criteria in related work.

Literature	Severity Score	C	I	A	Age/Time	CCE	CDP	ER	VT
Al-Ayed et al. [29]	✓							✓	✓
Walkowski et al. [23]	✓	✓	✓	✓			✓		
Kamongi et al. [30]	✓	✓	✓	✓	✓			✓	
Araujo and Taylor [31]	✓					✓		✓	✓
Fruhwrith and Mannisto [18]	✓	✓	✓	✓				✓	
Patil and Modi [32]	✓	✓	✓	✓			✓	✓	
Lee et al. [33]	✓	✓	✓	✓	✓				✓
Angelini et al. [34]	✓				✓			✓	
Lin et al. [35]	✓	✓	✓	✓		✓	✓		
Li et al. [36]	✓	✓	✓	✓				✓	
Torkura et al. [37]	✓				✓				
Yadav et al. [24]	✓	✓	✓	✓				✓	
Olswang et al. [38]	✓							✓	✓
Jiang et al. [26]	✓	✓	✓	✓		✓		✓	
Shah et al. [25]	✓	✓	✓	✓	✓			✓	

5.2. Experts' Interview

We interviewed nine vulnerability-management experts from governmental and private sectors located in the USA and EU. The experts who participated in the study worked in the information-technology domain with different levels of experience; (a) three juniors who have less than two years of experience in VRM; (b) three middle-level employees who have from two to five years of experience in VRM; and (c) three seniors who have more than five years of experience in VRM. We chose three different seniority levels as the response depends on knowledge and experience level [41]. The interview included two parts. In the first part, we interviewed the experts regarding the process they used to evaluate patch prioritization in their organization. In the second part, we asked the experts to rank the metrics in CVSS V3 and the accessibility level of the vulnerable AUS. The VRM experts ranked the following criteria in the second part of the interview:

- Attack vector (AV) is a CVSS V3 exploitability metric which refers to the context of the possibility of vulnerability exploitation (i.e., exploit vulnerability component from a network or locally)
- Attack complexity (AC) is a CVSS V3 exploitability metric which defines the condition that must exist in the environment to exploit the vulnerability. For example, if any security controls do not protect the vulnerable component, the attacker could successfully exploit the vulnerability with less effort.
- Privilege requirements (PR) is a CVSS V3 exploitability metric which describes the level of privilege an attacker must have to exploit the vulnerability successfully.
- User interaction (UI) is a CVSS V3 exploitability metric which expresses the human intervention in the successful comprise of the vulnerable component.
- Confidentiality (C) is a CVSS V3 impact metric which measures the impact on the confidentiality of the source after a successful attack.
- Integrity (I) is a CVSS V3 impact metric which measures the impact on the integrity of the source after a successful attack.
- Availability (A) is a CVSS V3 impact metric which measures the impact on the availability of the source after successful exploitation.
- Severity score (SC) is an output of CVSS which captures the technical characteristics of a component to a numeric score indicating the severity of the vulnerability.
- Internal AUS refers to the services that are not exposed to the public.
- External AUS refers to the services that are exposed to the public.

The interview was conducted in a virtual session on Microsoft Teams for around 60 min. Naturally, the number of experts in any domain is limited, which affects the number of available expert participants. Therefore, experts' participation in any study is lower than non-expert participants. Isenberg et al. [42] found the median number of expert participants in the study is nine in a survey of 113 papers.

The interviewees' ranked the criteria from one to five, where one is the lowest and five is the highest. From the interview, we calculated the statistics, including the minimum, maximum, average, and standard deviation of the expert's score in Table 3. In Figure 4, we show the individual experts' feedback. Looking at the statistics, we see that external AUS is the criterion with the highest average rating of 4.67. However, C, AV, I, A, and SC are also rated above 4.

Table 3. Statistics of the criteria ranked by security experts

Statistics	AV	AC	PR	UI	C	I	A	SC	Internal AUS	External AUS
Minimum	3	3	2	2	4	4	3	3	1	4
Maximum	5	4	3	4	5	5	5	5	3	5
Average	4.56	3.33	2.78	2.89	4.67	4.44	4.33	4.33	2.11	4.78
Standard deviation	0.68	0.47	0.42	0.74	0.47	0.50	0.67	0.67	0.57	0.42

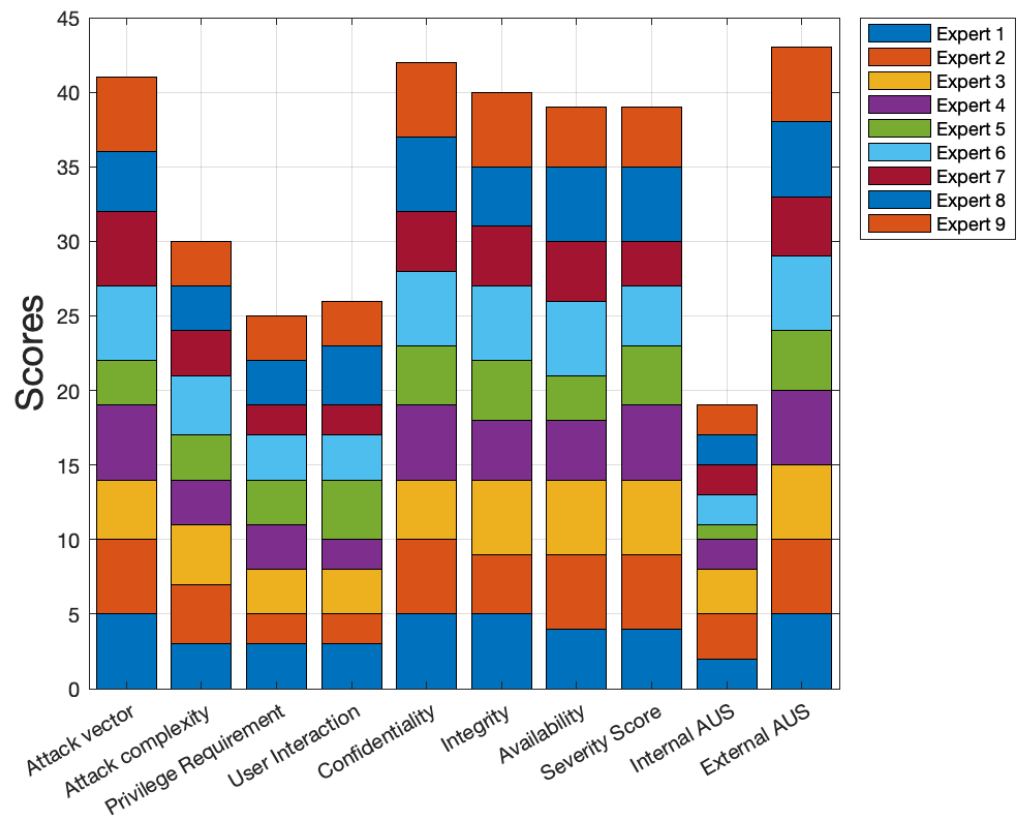


Figure 4. Ranking of the criteria by security experts.

5.3. Selected Criteria

We analyzed the related work and expert interview results to identify criteria with an average score above 3 (e.g., above 60% of maximum scores by experts and above 60% of literature). Based on the results in Tables 2 and 3, we chose the SC, C, I, A, AV, AC, and external AUS. The selected criteria, except external AUS, have defined metrics in the CVSS framework. Therefore, we can retrieve these from the CVSS vector or the vulnerability description reported by VDs. Some VDs, such as NVD and RHSA, report the CVSS vector, and some, such as USN and DSA, use similar keywords in the vulnerability description. In this study, we use the CVSSv3.1 vector to retrieve the score of selected criteria and calculate the patch score, cf. Section 5.4. The external AUS information the OC provides in access policy and group-based services.

5.4. Patch Score (PS)

ACVRM uses a patch score (PS) to determine the patch priority. PS is a scaling factor which can amplify the severity score of each CVE, and it is a function of the evaluation criteria as defined in Equation (1)

$$PS_k = \sum_{i=1}^n w_i F_{k,i} + \begin{cases} 2 & \text{for } AV = N \text{ and } AC = L \\ 1 & \text{for } AV = N \text{ and } AC = H \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where: PS_k is the PS for vulnerability k ; n is the number of evaluation criteria considered; w_i is a weight such that $w_i \in [0, 1]$, and $\sum_{i=1}^n w_i = 1$; $F_k = [F_{k,1}, \dots, F_{k,n}]$ is the impact vector for vulnerability k . In this paper, we use $n = 6$ and $F_k = [SC_k, AV_k, AC_k, C_k, I_k, A_k]$. F_k can be easily expanded or reduced depending on the criteria considered.

Equation (1) amplifies the PS for the vulnerabilities that could be exploited from network $AV_k = N$. The PS_k increases by additive factor +2 for vulnerability k with a low

complexity ($AC_k = L$). The PS_k raises by the additive factor +1 when the $AC_k = H$ is high. In all the other cases, i.e., the attack is not exposed to a network, no amplification is added.

To calculate a PS , we need to retrieve the weight vector from OC and the criteria vector from the CVSS vector in our local record. The organization could weigh each criterion based on its importance and influence the PS value. The CVSS vector has been available since 2000 in NVD, and the chance of not having CVSS vector information is negligible. CVSS vector is a data string that captures the corresponding value for each CVSS metric. The CVSS vector, e.g.,

CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N

starts with CVSS as a key and the version of CVSS (here: 3.1) as a value. The forward slash is a delimiter between each metric. The abbreviation of each base metric is used as a key and separated from the abbreviated metric's value with a colon.

Table 4 presents the metrics' name and metrics' value and their abbreviation from CVSS v3.1 document [16]. The CVSS base metric groups consist of AV, AC, PR, UI, Scope (S), C, I, and A. We excluded the PR, UI, and S metrics as they were not selected to consider in this study.

By expanding Equation (1) into the form used in this paper, we obtain:

$$PS_k = w_1 SC_k + w_2 AV_k + w_3 AC_k + w_4 C_k + w_5 I_k + w_6 A_k + \begin{cases} 2 & \text{for } AV_k = N \text{ and } AC_k = L \\ 1 & \text{for } AV_k = N \text{ and } AC_k = H \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The PS value will be between 0 and 12. The maximum PS value could be achieved when $SC = 10$ and $w_1 = 1$ and $AV = N$ and $AC = L$. The PS score could be zero when the highest weight is given to the metric, which happens to be none.

Table 4. CVSS v3.1 metric and the value used in ACVRM patch score [16].

Metric Name	Metric Value	Numeric Value
Attack Vector (AV)	Network (N)	0.85
	Adjacent (A)	0.62
	Local (L)	0.55
	Physical (P)	0.2
Attack complexity (AC)	Low (N)	0.77
	High (H)	0.44
Confidentiality (C)	None (N)	0
	Low (L)	0.22
	High (H)	0.56
Integrity (I)	None (N)	0
	Low (L)	0.22
	High (H)	0.56
Availability (A)	None (N)	0
	Low (L)	0.22
	High (H)	0.56

6. Design and Implementation

This section describes the implementation of a PoC for the prioritization phase (phase 2) of ACVRM, shown in Figure 5. It is designed as a group of functions split into four modules. Each module represents the implementation of each stage in phase 2,

and the output of each module is the input for the next one. We chose JSON as the internal data representation in this implementation since it is a supported format for most VDs and inventory tools.

Phase 1 PoC was described in [7]; hence, we do not repeat it here. The output from it is a file `NF_output.json`. This contains the CVE-ID, normalized scores for CVE, name of vulnerable AUS, safe version of AUS, and the severity score from one or multiple VDs.

The filter module matches the name of vulnerable AUS in the `NF_output.json` and the AUS information in the host inventory (`host_inventory.json`) to detect the organization's vulnerabilities. It also adds the vulnerabilities reported by the vulnerability scanner (`vulnerability_scanner_report.json`) if that is not already identified in `host_inventory.json`. The organization could set a VMM in the VMM file as a default. The host-based VMM is an alternative for the organization if needed, and the VMM value should be added to the host inventory. The output of the filter stage is the `filter.json`, which consists of CVE-ID, a normalized score for the CVE-ID, hosts name, and the vulnerable AUS.

The evaluation module assesses each entry (i.e., CVE-ID) in the `filter.json` based on the access policy and group host-based services to separate internal AUS and external AUS. Then, we implement the check on the selected criteria and their weight to calculate the PS for each CVE-ID. If the organization does not provide the weight vector, ACVRM will weigh all criteria equally. The evaluation output is the `external_list.json` and `internal_list.json`, which refer to the vulnerabilities affecting internal and external AUS. Each list provides the CVE-ID, normalization score, CVSS vector, PS, host name and group (if applicable), the name of AUS, and the original severity score from VDs for each CVE-ID.

The sort module provides patch prioritization influenced by service priority. The `servicepriority.json` is a list of critical services for an organization's business. Hence, vulnerability remediation on those services should receive the highest priority. If the organization does not have preferences, the patch prioritization will be based on the PS value computed from Equation (1).

The output of phase 2 of ACVRM is the patch priority for each host in the inventory. The internal and external AUS are sorted separately, as the patch time might differ for each group. This output feeds into the patch-management tool in phase 3 of ACVRM.

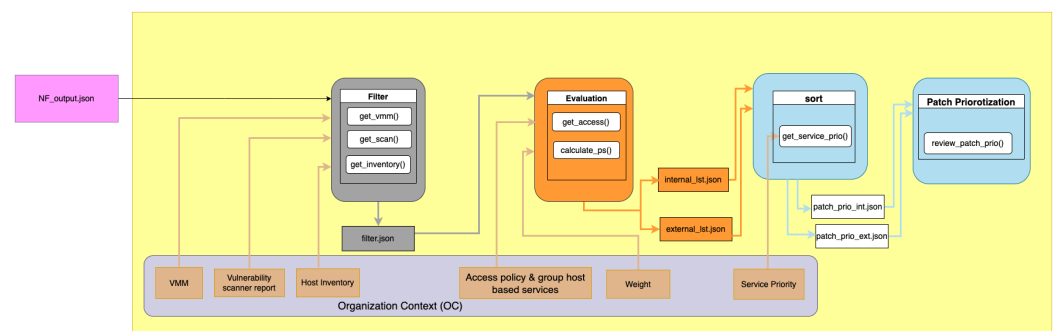


Figure 5. High-level software architecture of phase 2—prioritization.

7. Experimental Validation of PoC

The setup of the experimental environment to deploy and validate the ACVRM's PoC is organized into four parts.

First, we collect the CVE-IDs data from VDs corresponding to phase 1, described in Section 7.1. In the second stage, we create a virtual company. This company's organizational environment is characterized by a network of virtual servers deployed on a public cloud platform. This setup is shown in Figure 6; it consists of nine virtual servers (Ubuntu1-3, Debian1-3, CentOS1-3), one storage node (local storage), one Rudder node, and one Nessus node. All nodes are connected to a switch. The servers are organized into three groups of three nodes each, where each group runs a different Linux distribution. Rudder node is a host running the Rudder.io manager version 6.2 [43] as an inventory tool. The Rudder

manager receives the nodes' data through the installed Rudder agent on the nine virtual servers. The Nessus node is a host running the Nessus [5] vulnerability scanner community edition, version 8.14.0-ubuntu110_amd64. The community edition of Nessus does not provide the CVE-ID of the detected vulnerabilities but instead reports the vulnerability with the Nessus ID. The report is generated in a limited format, such as HTML and CSV, and does not support Rest API. The report should be converted to JSON with the corresponding CVE-ID.

The nodes are created using the OS image provided by the cloud provider and then updated to the latest stable version. Table 5 describes the nodes' specifications.

In the third stage, we deploy the prioritization stage of ACVRM in our test environments to obtain the patch-priority list for a given organization with four cases; each has different weight vectors. Finally, the fourth stage compares the output of the ACVRM prioritization of each case with the Rudder.io's CVE plugin [44] results. CVE plugin is a VRM software developed by Rudder.io to identify and prioritize the vulnerabilities in installed software on each node managed by Rudder.

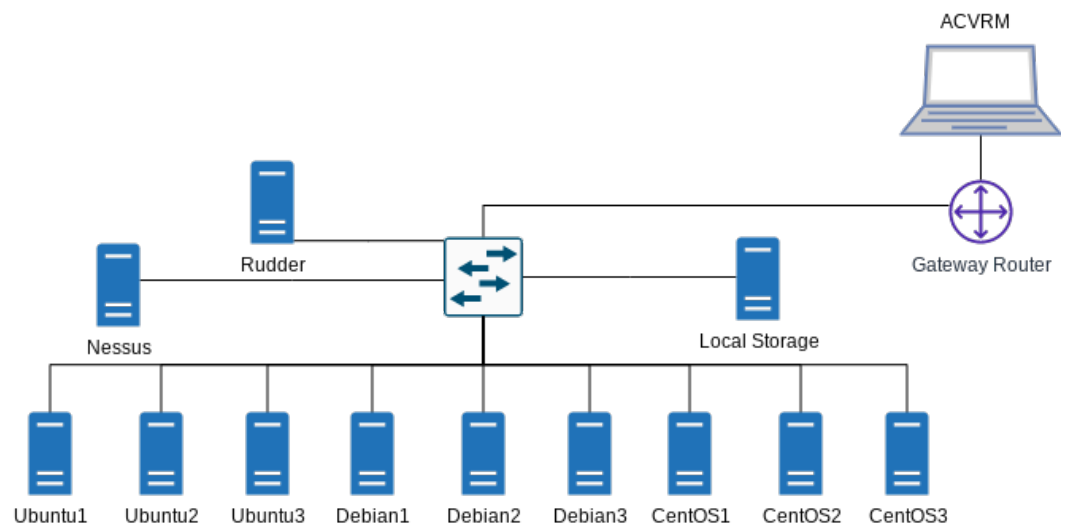


Figure 6. The test environments

Table 5. The specifications for the virtual servers in the test environment.

Specification	Ubuntu	Debian	Centos	Rudder	Nessus	Storage
CPU		2		8	4	4
RAM		2 GB		16 GB	4 GB	4 GB
Storage		20 GB		50 GB	20 GB	1 TB
Distribution	18.04.4 LTS	9.1.1	8.1.1911	18.04.4 LTS	18.04.4 LTS	18.04.4 LTS
Kernel	4.15.0-158-generic	4.9.272-2	4.18.0-305.10.2.el8_4	4.15.0-158-generic	4.15.0-158-generic	4.15.0-158-generic
Nodes	3	3	3	1	1	1

7.1. Phase 1: Data Collection and Pre-Processing

We collect CVE-IDs data from the four VDs described below, this is based on the existing OS in our experiment, and we add NVD as a reference:

- RedHat Security Advisory (RHSA) [9] is a subject-specific VD that provides the severity score based on the base and environmental metrics of CVSS v3.x. RHSA records the severity score for the CVE-IDs that effects RedHat's releases. RHSA reports the quantitative score and the severity rating based on the impact of the vulnerability in RedHat environments.
- Ubuntu Security Notices (USN) [10] is a subject-specific VD that reports the CVE-IDs affected by Ubuntu's releases. USN developed its framework for calculating severity

scores that are not publicly available. USN provides a qualitative severity score for each CVE-ID.

- Debian Security Advisories (DSA) [45] is a subject-specific VD that records the CVE-IDs affected by Debian's releases. DSA delivers a qualitative severity score, which relies on the NVD score. It is not clear which version of CVSS is applied to the DSA score.
- National Vulnerability Database (NVD) [8] provides a severity score based on a base metric of the CVSS v2.0 and v3.x framework. NVD is a generic VD and does not consider the environmental metrics in severity scores. NVD is one of the largest VD that records almost all existing CVE-IDs.

The data collected from the four VDs mentioned above are related to the CVEs affecting Linux distributions from 2017 to 2021 for this study. These raw data are kept in our storage node in the JSON format as a reference. The data is collected daily and archived in our local storage node, independent of other stages. This study is based on the information collected in June 2022.

7.2. Phase 2: Prioritization

Phase 2 implements the identification, classification, and evaluation processes of VRM. This phase automatically processed the data from phase 1 of ACVRM regarding OC. In our experiment, the Rudder node provides host inventory data, and the Nessus node generates the vulnerability scan report of the nine virtual servers. Then, we create a group of host-based services for the organization regarding the operating systems. One host from each host group is configured as an external AUS and exposed to the Internet. The rest of the virtual servers are set up as internal AUS. Finally, we assign public IP addresses to the external AUS and configured an SMTP server on them. Our test organization does not prioritize services and operates on standard VMM but with four different weight vectors. After preparing our test organization, we update all nine virtual servers to patch all existing vulnerabilities via the Rudder CVE plugin. We scan with Nessus and run the CVE plugin in check mode to validate that vulnerabilities are successfully patched in servers.

We randomly selected 24 CVE-IDs (relevant to our virtual servers) and installed the vulnerable version on our nine virtual servers (Ubuntu1-3, Debian1-3, and CentOS1-3). Table 6 shows the selected CVE-IDs, the name of AUS, the severity score in each selected VD in our experiment, the CVSS vector for each CVE-ID, and the normalized score for each CVE-ID with standard VMM. We started our experiment by manually executing the Rudder and Nessus to determine the detected installed vulnerabilities. Then, we run the code for phase 2 PoC. The stages in phase 2 are implemented in Python and comprise a group of functions. The functions' execution should be in order according to Figure 5 because the output of each stage is an input for the next one in the group. We keep the state of the virtual servers (e.g., with 24 installed vulnerabilities) unchanged and review the patch-prioritization list provided by our tool for each case. We repeated the execution with standard VMM 50 times to verify that our codes gave an identical result. The execution time for phase 2 PoC was 7 min in our test environment. However, our focus in this experiment was on the accuracy of the patch prioritization rather than the execution time. In the future, we will investigate the relationship between execution time and the number of nodes.

Table 6. The sample of vulnerabilities that affect our test environments.

CVE-ID	AUS Name	SC RHSA	SC DSA	SC USN	SC NVD	CVSS Vector	Normalized Score
CVE-2021-33574	glibc	5.9	high	low	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	6.4125
CVE-2021-3796	vim	7.3	medium	medium	7.3	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:H	6.7500
CVE-2021-4192	vim	7.8	medium	medium	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	6.6250
CVE-2021-3778	vim	7.8	medium	medium	7.8	CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H	6.6250
CVE-2021-22555	kernel	7.8	medium	high	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	7.2500
CVE-2020-28374	kernel	8.1	medium	high	8.1	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	7.4000
CVE-2021-4034	polkit	7.8	high	high	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	7.2500
CVE-2021-22946	curl	7.5	medium	medium	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	7.1000
CVE-2021-3712	openssl	7.4	medium	medium	7.4	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:H	6.4250
CVE-2021-3449	openssl	5.9	medium	high	5.9	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H	6.3000
CVE-2021-41617	openssh	7.0	medium	low	7.0	CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H	5.3625
CVE-2020-13776	systemd	6.7	medium	low	6.7	CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:U/C:H/I:H/A:H	5.2125
CVE-2021-33910	systemd	5.5	medium	high	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H	5.2375
CVE-2020-14308	grub2	6.4	medium	high	6.4	CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H	6.5500
CVE-2021-30465	runc	7.5	medium	high	8.5	CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:C:H/I:H/A:H	7.3500
CVE-2021-20277	libldb	7.1	medium	high	7.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	7.0000
CVE-2020-8831	appopt	None	low	high	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N	5.1500
CVE-2020-8794	openSMTPD	None	high	high	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	8.5667
CVE-2021-3177	python	5.9	high	medium	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	7.2750
CVE-2021-20179	dogtag-pki	8.1	medium	high	8.1	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	7.4000
CVE-2021-27135	xterm	9.6	high	medium	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	8.2000
CVE-2021-3156	sudo	7.8	high	high	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	7.8750
CVE-2020-11651	salt	9.8	high	medium	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	8.2500
CVE-2021-32760	containerd	5.5	medium	high	6.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L	6.3000

8. Results and Discussion

This section analyzes the patch-prioritization result generated by our tool for each case and Rudder CVE-Plugin. We used the PS value to prioritize the patch order. As described in Section 5.4, the PS value captures the important criteria in ranking the vulnerabilities. The organization could weigh the criteria based on risk appetite to customize patch prioritization. This study considers four cases with different weighting criteria to study patch prioritization with different risk appetites. Table 7 represents the numeric value of each criterion involved in our PS calculation. The SC column is the normalized score for each CVE-ID where the VMM is standard. The Add Factor column in Table 7 represents the additive value in Equation (1). The PS1-4 in Table 7 are the PS value for the case 1–4 respectively. The corresponding value of AV, AC, C, I, and A are from CVSS v3.1 and has two decimals. The result of our calculation, including SC, PS1, PS2, PS3, and PS4, are rounded to four decimals.

- Case 1: in this case, the organization weighs the criteria homogeneously in PS calculation as all six items are equally important for its business, i.e., $w_i = 1/N, \forall i$. Equation (3) is expanded from Equation (2) for each CVE-ID, i.e., k .

$$PS_k = 0.1667(SC_k + AV_k + AC_k + C_k + I_k + A_k) + \begin{cases} 2 & \text{for } AV_k = N \text{ and } AC_k = L \\ 1 & \text{for } AV_k = N \text{ and } AC_k = H \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

For example, the PS for $k = CVE - 2021 - 33574$ is calculated as:

$$PS_{CVE-2021-33574} = 0.1667(6.4125 + 0.85 + 0.77 + 0.56 + 0.56 + 0.56) + 2$$

The PS1 column in Table 7 shows the result for case 1 by ACVRM for each CVE-ID that affected our test environments.

- Case 2: The organization does not consider AC and A as important criteria in this case. Hence, $w_3 = w_6 = 0$. However, the organization weighs the rest of criteria homogeneously in PS calculation, i.e., $w_1 = w_2 = w_4 = w_5 = w_6 = 0.25$ and $\sum_{i=1}^6 w_i = 1$. Equation (4) is derived from Equation (2) for each CVE-ID, i.e., k , in case 2.

$$PS_k = 0.25(SC_k + AV_k) + 0 * AC_k + 0.25(C_k + I_k) + 0 * A_k + \begin{cases} 2 & \text{for } AV_k = N \text{ and } AC_k = L \\ 1 & \text{for } AV_k = N \text{ and } AC_k = H \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

For example, the PS for CVE-2021-33574 is calculated as:

$$PS_{CVE-2021-33574} = 0.25(6.4125 + 0.85) + 0 * 0.77 + 0.25(0.56 + 0.56) + 0 * 0.56 + 2$$

The result for case 2 by ACVRM is presetted in the PS2 column in Table 7.

- Case 3: the organization only considers the SC value for prioritizing the patch. Hence, the weight is distributed as $w_1 = 1$ and $w_2 = w_3 = w_4 = w_5 = w_6 = 0$. Equation (5) is obtained from Equation (2) for each CVE-ID, i.e., k , in case 3.

$$PS_k = 1 * SC_k + 0 * (AV_k + AC_k + C_k + I_k + A_k) + \begin{cases} 2 & \text{for } AV_k = N \text{ and } AC_k = L \\ 1 & \text{for } AV_k = N \text{ and } AC_k = H \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

For example, the PS for $k = CVE - 2021 - 33574$ is calculated as:

$$PS_{CVE-2021-33574} = 1 * 6.4125 + 0 * (0.85 + 0.77 + 0.56 + 0.56 + 0.56) + 2$$

Table 7 shows the outcome of case 3 for each CVE-ID in the PS3 column.

- Case 4: in this case, the organization weights all criteria based on its risk appetite. The weight is distributed as $w_1 = 0.35, w_2 = 0.2, w_3 = w_4 = w_6 = 0.1$, and $w_5 = 0.15$ in the PS calculation. Equation (6) is derived from Equation (2) for each CVE-ID, i.e., k , in case 4.

$$PS_k = 0.35 * SC_k + 0.2 * AV_k + 0.1(AC_k + C_k) + 0.15 * I_k + 0.1 * A_k + \begin{cases} 2 & \text{for } AV_k = N \text{ and } AC_k = L \\ 1 & \text{for } AV_k = N \text{ and } AC_k = H \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

For example, the PS for $k = CVE - 2021 - 33574$ is calculated as:

$$PS_{CVE-2021-33574} = 0.35 * 6.4125 + 0.2 * 0.85 + 0.1(0.77 + 0.56) + 0.15 * 0.56 + 0.1 * 0.56 + 2$$

The result of PS for case 4 is presented in column PS4 in Table 7 for the CVE-IDs that affected our test environments.

Analysing the Patch Prioritization

In this section, we compare the patch prioritization offered by Rudder’s CVE-plugin and four cases of ACVRM. Table 8 presents the patch order for different cases. Rudder CVE-plugin provided patch prioritization based on the SC from general purpose VD and NVD and does not reflect the organization’s context.

We define Δ as the difference between the position in patch priority ($P_{k,*}$) between Rudder and ACVRM cases as:

$$\Delta_k = P_{k,rudder} - P_{k,ACVRM} \quad (7)$$

where k is CVE-ID. The Δ column after each case in Table 8 shows the changes in the CVE-ID position compared with the Rudder CVE-plugin. For example, CVE-2021-33574 has a priority 1 by Rudder while it becomes priority seven in case 1 and priority eight in case 2. The Δ value with a negative sign means the position of the CVE-ID moves down (lower priority). In contrast, the positive value means the place of CVE-ID moves up (higher priority) in the priority list. The Δ is zero when the position of the CVE-ID is the same in the priority list provided by Rudder and ACVRM case. The CVE-ID with priority one will be patched first, and the CVE-ID with priority twenty-four in our list will be the last to be patched.

We observed that the CVE-IDs that could be exploited from networks with a low attack complexity gain higher priority (e.g., priority 1–11) by ACVRM compared with Rudder (e.g., priority 1–21). The priority position in Table 8 shows that only five CVE-IDs (e.g., CVE-2020-8794, CVE-2020-11651, CVE-2021-27135, CVE-2021-3449, CVE-2021-41617) were of the same priority in ACVRM cases.

After reviewing the Δ value, we found that CVE-2020-8831 obtains priority twenty-four in Rudder and ACVRM cases 1, 3, and 4. Case 2 of ACVRM does not have any similar priority position compared with Rudder. We also noticed that thirteen CVE-IDs achieved a lower priority position in Case 2 of ACVRM, while the number of CVE-IDs is eleven in other cases. In addition, we visualized the change in each vulnerability position for all cases in Figure 7. As shown in Figure 7, the position of five vulnerabilities (e.g., CVE-2021-33574, CVE-2021-27135, CVE-2020-8794, CVE-2021-4034, CVE-2020-14308) rise for all four cases where three of them (e.g., CVE-2021-33574, CVE-2021-27135, CVE-2020-14308) increased to the same order. We also observed that the position of six vulnerabilities (e.g., CVE-2021-3177, CVE-2021-3156, CVE-2021-22946, CVE-2021-20277, CVE-2021-41617, CVE-2021-32760) decreased in the priority list in all four cases while two of them (e.g., CVE-2021-32760, CVE-2021-3177) dropped to exactly the same order.

Table 7. Comparing the PS values in different cases with Rudder

CVE-ID	Rudder	SC	AV	AC	C	I	A	Add Factor	PS 1	PS 2	PS 3	PS 4
CVE-2021-33574	9.8	6.4125	0.85	0.77	0.56	0.56	0.56	2	3.6191	4.0956	8.4125	4.6874
CVE-2021-27135	9.8	8.2000	0.85	0.77	0.56	0.56	0.56	2	3.9171	4.5425	10.2000	5.3130
CVE-2021-3177	9.8	7.2750	0.85	0.77	0.56	0.56	0.56	2	3.7629	4.3113	9.2750	4.9893
CVE-2020-11651	9.8	8.2500	0.85	0.77	0.56	0.56	0.56	2	3.9254	4.5550	10.2500	5.3305
CVE-2020-8794	9.8	8.5667	0.85	0.77	0.56	0.56	0.56	2	3.9782	4.6342	10.5667	5.4413
CVE-2021-3796	8.8	6.7500	0.55	0.77	0.56	0.22	0.56	0	1.5686	2.0200	6.7500	2.6945
CVE-2021-30465	8.5	7.3500	0.85	0.44	0.56	0.56	0.56	1	2.7203	3.3300	8.3500	3.9825
CVE-2021-20179	8.1	7.4000	0.85	0.77	0.56	0.56	0	2	3.6903	4.3425	9.4000	4.9770
CVE-2020-28374	8.1	7.4000	0.85	0.77	0.56	0.56	0	0	3.6903	4.3425	9.4000	4.9770
CVE-2021-22555	7.8	7.2500	0.55	0.77	0.56	0.56	0.56	0	1.7087	2.2300	7.2500	2.9205
CVE-2021-4192	7.8	6.6250	0.55	0.77	0.56	0.56	0.56	0	1.6045	2.0738	6.6250	2.7018
CVE-2021-4034	7.8	7.2500	0.55	0.77	0.56	0.56	0.56	0	1.7087	2.2300	7.2500	2.9205
CVE-2021-3778	7.8	6.6250	0.55	0.77	0.56	0.56	0.56	0	1.6045	2.0738	6.6250	2.7018
CVE-2021-3156	7.8	7.8750	0.55	0.77	0.56	0.56	0.56	0	1.8129	2.3863	7.8750	3.1393
CVE-2021-22946	7.5	7.1000	0.85	0.77	0.56	0	0	2	3.5470	4.1275	9.1000	4.7880
CVE-2021-20277	7.5	7.0000	0.85	0.77	0	0	0.56	2	3.5303	3.9625	9.0000	4.7530
CVE-2021-3712	7.4	6.4250	0.85	0.44	0.56	0	0.56	1	2.4728	1.9588	7.4250	3.5748
CVE-2021-41617	7	5.3625	0.55	0.44	0.56	0.56	0.56	0	1.3390	1.7581	5.3625	2.2269
CVE-2020-13776	6.7	5.2125	0.55	0.44	0.56	0.56	0.56	0	1.3140	1.7206	5.2125	2.1744
CVE-2020-14308	6.4	6.5500	0.55	0.44	0.56	0.56	0.56	0	1.5370	2.0550	6.5500	2.6425
CVE-2021-32760	6.3	6.3000	0.85	0.77	0.22	0.22	0.22	2	3.4303	3.8975	8.3000	4.5290
CVE-2021-3449	5.9	6.3000	0.85	0.44	0	0	0.56	0	1.3586	1.7875	6.3000	2.4750
CVE-2021-33910	5.5	5.2375	0.55	0.77	0	0	0.56	0	1.1865	1.4469	5.2375	2.0761
CVE-2020-8831	5.5	5.1500	0.55	0.77	0	0.56	0	0	1.1719	1.5650	5.1500	2.0735

Table 8. Patch priority list in test environments by Rudder CVE-plugin and ACVRM.

Priority	Rudder		Case 1		Case 2		Case 3		Case 4	
	CVE-ID	CVE-ID	Δ	CVE-ID	Δ	CVE-ID	Δ	CVE-ID	Δ	
1	CVE-2021-33574	CVE-2020-8794	4	CVE-2020-8794	4	CVE-2020-8794	4	CVE-2020-8794	4	
2	CVE-2021-27135	CVE-2020-11651	2	CVE-2020-11651	2	CVE-2020-11651	2	CVE-2020-11651	2	
3	CVE-2021-3177	CVE-2021-27135	-1	CVE-2021-27135	-1	CVE-2021-27135	-1	CVE-2021-27135	-1	
4	CVE-2020-11651	CVE-2021-3177	-1	CVE-2020-28374	5	CVE-2020-28374	5	CVE-2021-3177	-1	
5	CVE-2020-8794	CVE-2020-28374	4	CVE-2021-20179	3	CVE-2021-20179	3	CVE-2020-28374	4	
6	CVE-2021-3796	CVE-2021-20179	2	CVE-2021-3177	-3	CVE-2021-3177	-3	CVE-2021-20179	2	
7	CVE-2021-30465	CVE-2021-33574	-6	CVE-2021-22946	8	CVE-2021-22946	8	CVE-2021-22946	8	
8	CVE-2021-20179	CVE-2021-22946	7	CVE-2021-33574	-7	CVE-2021-20277	8	CVE-2021-20277	8	
9	CVE-2020-28374	CVE-2021-20277	7	CVE-2021-20277	7	CVE-2021-33574	-8	CVE-2021-33574	-8	
10	CVE-2021-22555	CVE-2021-32760	11	CVE-2021-32760	11	CVE-2021-30465	-3	CVE-2021-32760	11	
11	CVE-2021-4192	CVE-2021-30465	-4	CVE-2021-30465	-4	CVE-2021-32760	10	CVE-2021-30465	-4	
12	CVE-2021-4034	CVE-2021-3712	5	CVE-2021-3156	2	CVE-2021-3156	2	CVE-2021-3712	5	
13	CVE-2021-3778	CVE-2021-3156	1	CVE-2021-4034	-1	CVE-2021-3712	4	CVE-2021-3156	1	
14	CVE-2021-3156	CVE-2021-4034	-2	CVE-2021-22555	-4	CVE-2021-4034	-2	CVE-2021-4034	-2	
15	CVE-2021-22946	CVE-2021-22555	-5	CVE-2021-3778	-2	CVE-2021-22555	-5	CVE-2021-22555	-5	
16	CVE-2021-20277	CVE-2021-3778	-3	CVE-2021-4192	-5	CVE-2021-3796	-10	CVE-2021-3778	-3	
17	CVE-2021-3712	CVE-2021-4192	-6	CVE-2020-14308	3	CVE-2021-3778	-4	CVE-2021-4192	-6	
18	CVE-2021-41617	CVE-2021-3796	-12	CVE-2021-3796	-12	CVE-2021-4192	-7	CVE-2021-3796	-12	
19	CVE-2020-13776	CVE-2020-14308	1	CVE-2021-3712	-2	CVE-2020-14308	1	CVE-2020-14308	1	
20	CVE-2020-14308	CVE-2021-3449	2	CVE-2021-3449	2	CVE-2021-3449	2	CVE-2021-3449	2	
21	CVE-2021-32760	CVE-2021-41617	-3	CVE-2021-41617	-3	CVE-2021-41617	-3	CVE-2021-41617	-3	
22	CVE-2021-3449	CVE-2020-13776	-3	CVE-2020-13776	-3	CVE-2021-33910	1	CVE-2020-13776	-3	
23	CVE-2021-33910	CVE-2021-33910	0	CVE-2020-8831	1	CVE-2020-13776	-4	CVE-2021-33910	0	
24	CVE-2020-8831	CVE-2020-8831	0	CVE-2021-33910	-1	CVE-2020-8831	0	CVE-2020-8831	0	

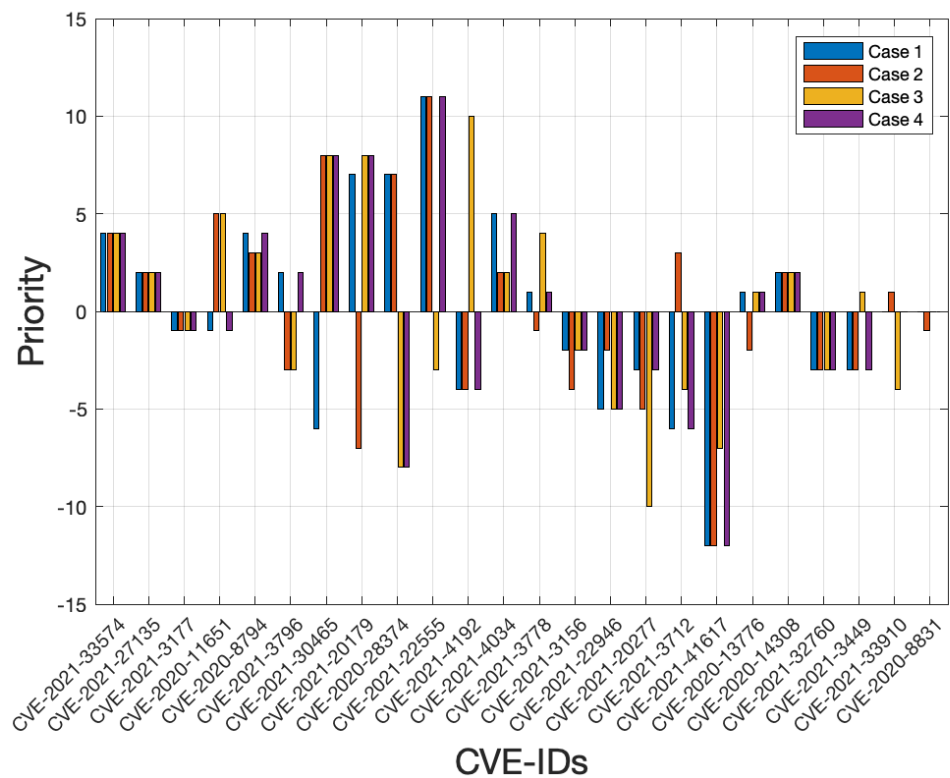


Figure 7. The Δ value of patch-priority orders in Rudder CVE-plugin and Case 1-4 of ACVRM.

We also noticed that Rudder listed the CVE-IDs with the same SC randomly (e.g., priority 1 to 5 have a SC = 9.8, and the position is not related to the age of the CVE-IDs). However, ACVRM considered the age of the CVE-IDs in prioritization when the PS value is

equal (e.g., CVE-2020-28374 and CVE-2021-20179 in case 1 have the same PS value but the CVE-2020-28374 gains the higher priority as it has been known publicly for a longer time).

9. Conclusions

The increasing number of publicly known vulnerabilities introduces a challenge to VRM as classification and evaluation phases need experts' intervention. Security experts should evaluate the vulnerability risk for organizations and define patch prioritization, which is time-consuming and resource-intensive. Therefore, we need to improve VRM to address the challenges mentioned earlier. We introduce ACVRM to automate the VRM procedure and reduce experts' intervention. Hence, we need to learn how experts evaluate and prioritize patching. In this study, we focus on the classification and evaluation process of VRM in the context of a given organization. We performed an analysis in three phases as follows:

1. We conducted a literature study and expert interviews to learn which criteria play a role in patch prioritization. We defined the selected criteria for patch prioritization based on the result obtained in our study. We found that the security score, attack vector, attack complexity, confidentiality, integrity, and availability values and exposure level of the AUS are essential in deciding the patch order. Therefore, we define the PS based on the selected criteria and the possibility of weighting each criterion in the organization's context.
2. We designed and implemented phase 2 of ACVRM, which consists of four modules: filter, evaluation, sort, and patch prioritization. We created the environments of the test organization in the public cloud. The experiment was executed for four cases where each case's criteria were weighted differently.
3. We verified the result of our phase 2 implementation by analyzing the outcome of each case. We also compared the patch prioritization of our tool with the Rudder CVE-plugin. Our result shows that the ACVRM could adjust the patch prioritization for each organization with less effort from security experts. The security experts only set the VMM and weigh the selected criteria. Our solution also allows the security experts to add more criteria to the evaluation module if needed.

Our study showed how an organization could customize patch priority based on its context by selecting VMM mode and weighting the criteria. We presented the improvement in the VRM procedure by reducing evaluation time and experts' intervention. The execution time of the phase 2 was seven minutes in our test environment, including four modules (e.g., filter, evaluation, sort, and patch prioritization). However, the execution time needs to be studied further.

In the future, we want to continue the implementation of phase 3 of ACVRM and address the challenges in patch management, including the automated validation of the patch deployment, verification of the side effects of patching vulnerabilities, and possibility of a generalized verification process. Another possible future direction could be using a machine-learning algorithm to improve patch prioritization based on the patch verification feedback. Finally, we could investigate the time efficiency of our solution and compare the patch prioritization of our proposed solution with the recently published state-of-the-art approaches.

Author Contributions: Conceptualization, V.A.M., P.A. and E.C.; methodology, V.A.M., P.A. and E.C.; software, V.A.M.; validation, V.A.M., P.A. and E.C.; formal analysis, V.A.M., P.A. and E.C.; investigation, V.A.M.; resources, E.C.; data curation, V.A.M.; writing—original draft preparation, V.A.M., P.A. and E.C.; writing—review and editing, V.A.M., P.A. and E.C.; visualization, V.A.M., P.A. and E.C.; supervision, E.C. and P.A.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A	Availability
AC	Attack complexity
ACVRM	Automated context-aware vulnerability risk management
AUS	Application/unit/service
AV	Attack vector
C	Confidentiality
CIS	Center of internet security
CVE	Common vulnerabilities and exposures
CVSS	Common vulnerability scoring system
DSA	Debian Security Advisories
I	Integrity
NVD	National Vulnerability Database
OC	Organization context
PoC	Proof of concept
PS	Patch score
RHSA	RedHat Security Advisory
SC	Severity score
USN	Ubuntu Security Notice
VD	Vulnerability database
VMM	Vulnerability management mode
VRM	Vulnerability risk management

References

1. Top Routinely Exploited Vulnerabilities. Available online: <https://www.cisa.gov/uscert/ncas/alerts/aa22-117a> (accessed on 12 June 2022).
2. Costs and Consequences of Gaps in Vulnerability Response. Available online: <https://www.servicenow.com/lpayr/ponemon-vulnerability-survey.html> (accessed on 26 August 2022).
3. *Vulnerability and Threat Trends Report 2021*; Technical Report; SkyBox Security, 2021. Available online: [https://www.skyboxsecurity.com/resource-library/?resource_search=&resource_type\[\]=report](https://www.skyboxsecurity.com/resource-library/?resource_search=&resource_type[]=report) (accessed on 18 October 2022).
4. Open Vulnerability Assessment Scanner (OpenVAS). Available online: <https://www.openvas.org/> (accessed on 18 October 2022).
5. Nessus Vulnerability Scanner. Available online: <https://www.tenable.com/products/nessus> (accessed on 18 October 2022).
6. Ahmadi, V.; Arlos, P.; Casalicchio, E. Normalization of Severity Rating for Automated Context-aware Vulnerability Risk Management. In Proceedings of the 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), Washington, DC, USA, 17–21 August 2020.
7. Ahmadi, V.; Arlos, P.; Casalicchio, E. Normalization Framework for Vulnerability Risk Management in Cloud. In Proceedings of the 2021 IEEE International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 9 November 2021.
8. NIST National Vulnerability Database. Available online: <https://nvd.nist.gov/> (accessed on 15 October 2022).
9. RedHat Security Advisories. Available online: <https://access.redhat.com/security/security-updates/#/> (accessed on 10 October 2022).
10. Ubuntu Security Notice. Available online: <https://usn.ubuntu.com/> (accessed on 8 September 2022).
11. Apache Security Information. Available online: <https://www.apache.org/security/projects.html> (accessed on 16 September 2022).
12. CIS Controls . Available online: <http://www.cisecurity.org/controls/> (accessed on 10 October 2022).
13. EU Cybersecurity Act. Available online: <https://eur-lex.europa.eu/eli/reg/2019/881/oj> (accessed on 11 October 2022).
14. European Cybersecurity Certification Scheme for Cloud Services. Available online: <https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme> (accessed on 11 October 2022).
15. Homeland Security Act 2002. Available online: <https://www.dhs.gov/homeland-security-act-2002> (accessed on 15 October 2022).
16. Common Vulnerability Scoring System v3.1: Specification Document. Available online: <https://www.first.org/cvss/v3.1/specification-document> (accessed on 15 October 2022).
17. Spanos, G.; Sioziou, A.; Angelis, L. WIVSS: a new methodology for scoring information systems vulnerabilities. In Proceedings of the 17th Panhellenic Conference on Informatics, Thessaloniki, Greece, 19–21 September 2013.
18. Fruhwirth, C.; Mannisto, T. Improving CVSS-based vulnerability prioritization and response with context information. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, 6 November 2009.

19. Cyber Security Report 2021 by Check Point Research. Available online: <https://www.checkpoint.com/downloads/resources/cyber-security-report-2021.pdf> (accessed on 16 October 2022).
20. Zhang, F.; Huff, P.; McClanahan, K.; Li, Q. A Machine Learning-based Approach for Automated Vulnerability Remediation Analysis. In Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 29 June 2020–1 July 2020.
21. Aota, M.; Kanehara, H.; Kubo, M.; Murata, N.; Sun, B.; Takahashi, T. Automation of Vulnerability Classification from its Description using Machine Learning. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020.
22. Wang, X.; Wang, S.; Sun, K.; Batcheller, A.; Jajodia, S. A Machine Learning Approach to Classify Security Patches into Vulnerability Types. In Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 29 June 2020–1 July 2020.
23. Walkowski, M.; Krakowiak, M.; Jaroszewski, M.; Oko, J.; Sujecki, S. Automatic CVSS-based vulnerability prioritization and response with context information. In Proceedings of the 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Hvar, Croatia, 23–25 September 2021.
24. Yadav, G.; Gauravaram, P.; Jindal, A.K.; Paul, K. SmartPatch: A patch prioritization framework. *Comput. Ind.* **2022**, *137*, 103595. [[CrossRef](#)]
25. Shah, A.; Farris, K.A.; Ganesan, R.; Jajodia, S. Vulnerability selection for remediation: An empirical analysis. *J. Def. Model. Simul.* **2022**, *19*. [[CrossRef](#)]
26. Jiang, Y.; Atif, Y. Towards automatic discovery and assessment of vulnerability severity in cyber–physical systems. *Array* **2022**, *15*, 100209. [[CrossRef](#)]
27. Common Vulnerabilities and Exposures (CVE). Available online: <https://cve.mitre.org/> (accessed on 18 October 2022).
28. Cloud Computing Compliance Criteria Catalogue (C5). Available online: https://www.bsi.bund.de/EN/Topics/CloudComputing/Compliance_Criteria_Catalogue/Compliance_Criteria_Catalogue_node.html (accessed on 18 October 2022).
29. Al-Ayed, A.; Furnell, S.; Zhao, D.; Dowland, P. An automated framework for managing security vulnerabilities. *Inf. Manag. Comput. Secur.* **2005**, *13*, 156–166. [[CrossRef](#)]
30. Zhang, F.; Li, Q. Dynamic Risk-Aware Patch Scheduling. In Proceedings of the 2020 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 29 June 2020–1 July 2020.
31. Araujo, F.; Taylor, T. Improving cybersecurity hygiene through JIT patching. In Proceedings of the Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, New York, NY, USA, 8–13 November 2020.
32. Patil, R.; Modi, C. Designing an efficient framework for vulnerability assessment and patching (VAP) in virtual environment of cloud computing. *J. Supercomput.* **2019**, *75*, 2862–2889. [[CrossRef](#)]
33. Lee, J.H.; Sohn, S.G.; Chang, B.H.; Chung, T.M. PKG-VUL: Security Vulnerability Evaluation and Patch Framework for Package-Based Systems. *ETRI J.* **2009**, *26*. [[CrossRef](#)]
34. Angelini, M.; Blasilli, G.; Catarci, T.; Lenti, S.; Santucci, G. Vulnus: Visual vulnerability analysis for network security. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 183–192. [[CrossRef](#)]
35. Lin, C.H.; Chen, C.H.; Lai, C.S. A study and implementation of vulnerability assessment and misconfiguration detection. In Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, Yilan, Taiwan, 9–12 December 2008.
36. Li, Z.; Tang, C.; Hu, J.; Chen, Z. Vulnerabilities Scoring Approach for Cloud SaaS. In Proceedings of the 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing and 2015 IEEE 12th International Conference on Autonomic and Trusted Computing and 2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), Beijing, China, 10–14 August 2015.
37. Torkura, K.A.; Cheng, F.; Meinel, C. A proposed framework for proactive vulnerability assessments in cloud deployments. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015.
38. Olswang, A.; Gonda, T.; Puzis, R.; Shani, G.; Shapira, B.; Tractinsky, N. Prioritizing vulnerability patches in large networks. *Expert Syst. Appl.* **2022**, *193*, 116467. [[CrossRef](#)]
39. Gusenbauer, M. Google Scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases. *Scientometrics* **2019**, *118*, 177–214. [[CrossRef](#)]
40. Common Configuration Enumeration (CCE). Available online: <https://ncp.nist.gov/cce/index> (accessed on 8 September 2022).
41. Zhang, D.C.; Wang, Y. An empirical approach to identifying subject matter experts for the development of situational judgment tests. *J. Pers. Psychol.* **2021**, *20*, 151–163. [[CrossRef](#)]
42. Isenberg, T.; Isenberg, P.; Chen, J.; Sedlmair, M.; Möller, T. A systematic review on the practice of evaluating visualization. *IEEE Trans. Vis. Comput. Graph.* **2013**, *19*, 2818–2827. [[CrossRef](#)] [[PubMed](#)]
43. Rudder. Available online: <https://www.rudder.io/> (accessed on 2 September 2022).
44. Rudder CVE Plugin. Available online: <https://docs.rudder.io/reference/6.2/plugins/cve.html> (accessed on 16 October 2022).
45. Debian Security Tracker. Available online: <https://www.debian.org/security/#DSAS> (accessed on 29 September 2022).