Normalization of Severity Rating for Automated Context-aware Vulnerability Risk Management

Vida Ahmadi
Dept. of Computer Science
Blekinge Institute of Technology
City Network International AB
Karlskrona, Sweden
email: vida.ahmadi@citynetwork.eu

Patrik Arlos
Dept. of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
email: patrik.arlos@bth.se

Emiliano Casalicchio
Dept. of Computer Science
Blekinge Institute of Technology, Sweden
Sapienza University of Rome, Italy
email: emiliano.casalicchio@bth.se

***Abstract*—In the last three years, the unprecedented increase in discovered vulnerabilities ranked with critical and high severity raise new challenges in Vulnerability Risk Management (VRM). Indeed, identifying, analyzing and remediating this high rate of vulnerabilities is labour intensive, especially for enterprises dealing with complex computing infrastructures such as Infrastructure-as-a-Service providers. Hence there is a demand for new criteria to prioritize vulnerabilities remediation and new automated/autonomic approaches to VRM.**

In this paper, we address the above challenge proposing an Automated Context-aware Vulnerability Risk Management (AC-VRM) methodology that aims: to reduce the labour intensive tasks of security experts; to prioritize vulnerability remediation on the basis of the organization context rather than risk severity only. The proposed solution considers multiple vulnerabilities databases to have a great coverage on known vulnerabilities and to determine the vulnerability rank. After the description of the new VRM methodology, we focus on the problem of obtaining a single vulnerability score by normalization and fusion of ranks obtained from multiple vulnerabilities databases. Our solution is a parametric normalization that accounts for organization needs/specifications.

***Index Terms*—vulnerability, automation, Risk Assessment, self-protection**

I. Introduction

VRM is a method to reduce the probability of exploitation and severity of damage in a system. Continuous VRM is ranked third of among critical security controls by the Center for Internet Security (CIS) [1]. Today, VRM is facing new complex challenges to remediate vulnerabilities, that are most critical for a specific organization context, on the proper time and sequence. The number of new vulnerabilities have increased enormously during the last three years, from 6,447 in 2016 to 17,306 unique vulnerabilities in 2019 [2]. According to the NIST National Vulnerability Database (NVD) [2], 57.17% of the new vulnerabilities reported in 2019 ranked with critical and high severity (41.82% high severity and 15.35% critical severity). Vulnerabilities with a higher probability of exploit need to be patched in a certain time-window regarding the organization context. For example, the federal agencies in U.S. have to patch the critical vulnerabilities within 15 days and the vulnerability with high severity within 30 days due to binding operational directive [1]. Hence, an autonomic VRM is vital to support decision makers by reducing labour intensive tasks and to address the large number of vulnerabilities within the specific deadlines.

Current VRM methods, used in production environments, prioritize vulnerability patching based on the severity score, thus vulnerabilities with critical and high severity are patched first. D. Dey et al. [3] compares several practical patch policies and concludes that patch policies relying on a single metric such as severity level are not optimal. Indeed, if the severity level of a vulnerability is high but the risk of exploitation is low, it can be patched in a later time. V. Katos et al. [4] reports that 8.65% of known vulnerabilities are exploitable, hence the current VRM practices should be changed to account for the exploitation probability, when prioritize patching.

Furthermore, the risk and impact of exploit of vulnerabilities depends on the organization's assets, security requirements and security policies (the organization context hereafter). Hence the vulnerabilities should not be analyzed in an isolated manner, but within the organization context; and also the risk of exploitation should be evaluated in the context of the organization to efficiently plan and to prioritize the vulnerability patching [5].

In this study, we address the aforementioned challenges by proposing AC-VRM. AC-VRM leverages knowledge about organization context to support a security decision-maker in prioritizing vulnerability patching. AC-VRM aims: 1) to reduce the labour intensive tasks of security experts; 2) to prioritize vulnerability remediation on the basis of the organization context rather than risk severity only; 3) it considers multiple vulnerabilities databases to have a great coverage on known vulnerabilities and to determine the vulnerability rank. AC-VRM is the first step towards the development of a self vulnerabilities assessment and remediation manager. The self assessment and remediation manager is shown in Fig. 1(b) and is compared with the self-protecting system manager for response and mitigation of cyber-attacks (widely investigated in literature [6]). A self-protecting system manager is a complex software that autonomously: monitor a system to collect information on its status; analyze the monitored data to detect anomalies (e.g. cyber-attacks or vulnerabilities); plan actions to bring the system back to normal operating conditions (e.g.

[1] https://cyber.dhs.gov/bod/19-02/

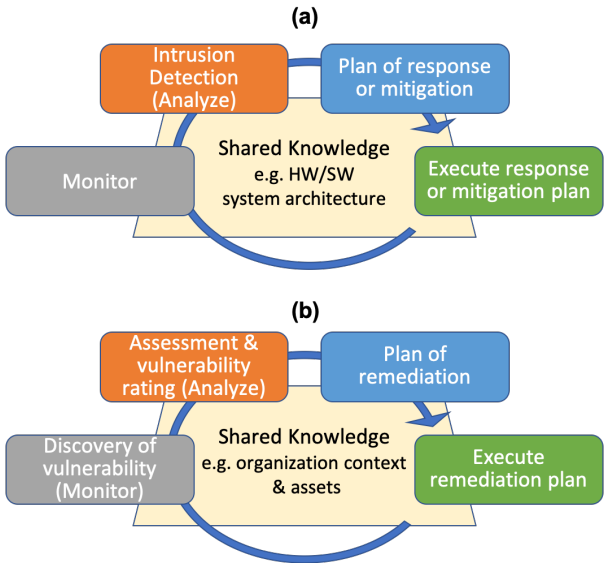

Fig. 1: (a) The self-response & mitigation manager; and (b) the self-vulnerability assessment & remediation manager.

attack response or mitigation actions, or vulnerabilities remediation actions); execute the planned actions. In this paper, we describe AC-VRM and we propose a normalization of vulnerability severity score based on the chosen vulnerability management mode by organization. That is we focus on the analysis phase in Fig. 1(b).

The remaining of this paper is organized as in what follows. Section II describes related scientific work and vulnerability management tools. The insight to VRM is described in Section III. Our proposed solution for an AC-VRM is presented in Section IV. Finally, Section V concludes the paper.

II. Related Work

In this section: firstly, we briefly review the literature about automated VRM and self-protection. Then, we describe the widely used tools for vulnerability management and we compare their features

A. Literature review

According with the taxonomy proposed in [6] AC-VRM could be classified as a self-protecting system that: 1) offers the planning and prevention self-protection level; 2) it is independent from the system layer to protect; 3) it addresses the confidentiality, integrity, and availability goals; and it focuses on development and deployment time.

Today, most research on self protecting systems mainly proposed solutions for self response and mitigation, e.g. [7], [8]. Only a few work has focused on self-vulnerability assessment and remediation [9].

Most of the literature on automated VRM address the problem of automated penetration testing (e.g. [10] [11]) or other form of automated vulnerability discovery (e.g. [12] [13] [14] [15]). In this work we assume that the organization has its own well tested methodology to discover vulnerabilities. Moreover, in complex production environments, like a Infrastructure-as-a-Service cloud provider, it is almost impossible to apply active vulnerability discovery methods [16].

In [17] the authors propose the Cyber Risk Scoring and Mitigation (CRISM) tool to estimate cyber-attack probabilities. CRISM directly monitors and scores cyber risk based on assets at risk and continuously updated software vulnerabilities. CRISM also produces risk scores that allows organizations to optimally choose mitigation policies that can potentially reduce insurance premiums. The approach used in CRISM is similar to the context-aware assessment proposed in our AC-VRM. However, CRISM uses only one database (NVD) as a source for known vulnerabilities. Moreover, the scoring is determined on the basis of a Byesian attack graph, that in case of a IaaS cloud provider could be extremely complex to generate and analyze.

In [18] the authors propose a framework for automated risk assessment and mitigation, that accounts for the user perceived risk. That proposed framework is tailored to a smart home system and does not take into account the problem of ranking and prioritizing vulnerabilities patching, like AC-VRM does.

In [15] the authors address the problem of zero-day vulnerabilities by proposing a method for risk assessment of zero-day vulnerabilities and attack vectors. Their method builds on a zero-day attack graph, analysis of pre- and post-conditions, and on the attack complexity score, and impact score obtained from the NVD database. As mentioned in the introduction and explained later in the paper, using only a single vulnerability database is a limitation. On the contrary, the limitation of the AC-VRM approach is to not consider explicitly zero-day vulnerabilities.

In [9] the authors propose a predictive machine learning model that can identify exploitable vulnerabilities, and that allows prioritization of patching by leveraging coverage (of vulnerabilities discovered) and efficiency (i.e. patching only what is at high exploitation risk for the organization). The machine learning model is trained on the vulnerabilities extracted from the CVE [19] database, while the NVD database is used to determine severity score. As we argue in this paper, considering only a single database to extract vulnerability score pose some risks, hence AC-VRM proposes to use different score and to normalize them on the basis of organization preferences (Vulnerability Management Mode, cf. Section IV).

B. Vulnerability Management Tools (VMT)

VMTs are used to discover vulnerabilities in network, software and hardware. Most of the VMTs are network vulnerability scanners [20]–[24]. However, some tools, cf. [25], rely on identification of vulnerabilities based on the system documentation. The output of VMT varies, but most generate a report. This report lists the detected vulnerabilities, and prioritizes based on their Common Vulnerability Scoring System (CVSS) score. Some of the more advanced VMTs include suggestions for mitigation, in the report [20]–[22].

Table I summarizes features of (some) common VMTs. That features are the following: *scan network*, i.e. the capability to scan the computer network of the organization; *identification*, that is the capability to detect existing vulnerabilities; *analysis*, i.e. the ability to rank the vulnerabilities by security score; *evaluation*, i.e. patching prioritization capacity; and *treatment*, that is the ability of patching the vulnerabilities (automatically or providing remediation plans). Supported features by tools are marked with a ✓, and the □ sign represents missing or limited support for a feature in Table I. From the table it is clear, that none of the tools provide a treatment that would immunize the systems by patching them. Furthermore, from our experience there is a lack of tools that can operate in a context of an organization and cover all aforementioned features.

III. Vulnerability Risk Management (VRM)

VRM is a procedure each organization should practice to maintain proactive cyber defence. Step one is identifying the vulnerability in the organization. The security team in an organization should regularly execute penetration testing (e.g. nmap) and vulnerability scanning (e.g. OpenVAS) on their systems to detect vulnerabilities and flaw in a system configuration, e.g. open ports and outdated software. The result of such a test is a list of detected vulnerabilities and their severity score, which is calculated by Vulnerability Database (VD). Therefore, VDs are a key components in any VRM method. The second step is analysing and ranking the detected vulnerabilities by using the security score obtained from the VD.

However, to properly and effectively analyze the detected vulnerabilities we need knowledge about the organization's security policy, exploitation probabilities, and impact of exploit on the system to plan the appropriate response. Considering only CVSS score to prioritize the patching mislead the security decision-maker. For example, a vulnerability with medium severity score, being actively exploited in the wild should be marked with a higher priority. As it might pose a greater risk, compare to a vulnerability with critical score, that has no known exploit [26]. Hence, the VRT process should be enhanced by adding a third step that prioritizes the patching and resolution of analyzed vulnerabilities based on the organization context.

Analyses and evaluation steps often rely on domain experts, who manually develop the protection mechanisms and define the time and order for patching. The evaluation of domain experts might leave unpatched some of the detected vulnerabilities due to the limited attack vector in the organization context or the high cost of patching compared to the exploit's cost. The aforementioned VRM process is resource intensive and slow due to the number of published and the rate that new vulnerabilities are discovered [27], [28]. Hence, the urgent demand for AC-VRM (or self vulnerability assessment and remediation controller).

The last step in VRM is patching the right vulnerabilities and verification of the procedure. As mentioned, the commonly used criteria in VRMs are the severity score of the vulnerability, CVSS described in the section III-B. The VDs are repositories that record the Common Vulnerabilities and Exposures (CVE) ID and provide a CVSS score for each CVE. Some of the VDs provide the remediation or workarounds to address that vulnerabilities.

A. Vulnerability Database (VD)

VDs are repositories that contain lists of publicly known vulnerabilities. VDs are usually hosted by organizations with certain regions of interests. For example, RedHat hosts a VD for tracking vulnerabilities affecting their products. However, thanks to CVE [19], each publicly known vulnerability is assigned an Identification Number (ID), the CVE ID. VDs creates entries in their internal system corresponding to the CVE IDs, based on the VDs' naming system, severity rating, and patch instruction. Therefore, the severity score might be different for a CVE ID in different VDs. Some of the commonly used VDs are described in what follows:

i) NIST National Vulnerability Database (NVD) is one of the largest vulnerability databases that contains all published CVEs. NVD operated by NIST as a part of the US Department of Commerce. NVD provides the security-related software flaw, effected products name, impact metrics and CVSS for each CVE [2].

ii) Ubuntu Security Notice (USN) collects all CVEs that affect different releases of Ubuntu. USN has its own naming method that uses four digits after USN as main ID followed by the version (i.e. USN-4295-1). The USN database provides a patch instruction, security score and CVE ID reference or references [29].

iii) RedHat Security Advisories (RHSA) reports the vulnerabilities that affect Redhat releases. Their naming system is the database name followed the year that vulnerability was published and four digits identifying the vulnerability in the database (i.e. RHSA-2020:2404). RHSA database describes each vulnerability, the affected release or releases, patch instruction, severity score, and the reference CVE [30].

iv) Cisco Security Advisories provides information about affected Cisco products by different CVEs. Cisco names each vulnerability by their product name and type of the exploit (i.e. Cisco IOS and IOS XE Software Tcl Denial of Service Vulnerability). Cisco Security Advisories describes each vulnerability and provides patch instruction, severity score, and reference CVE [31].

B. Severity Score

The Common Vulnerability Scoring System (CVSS) was created by FIRST.org, Inc. [32] and it aims to transfer the vulnerability characteristics to a numeric score, and help with the quantitative analysis in vulnerability management. CVSS is an open framework to communicate attributes and the severity for each vulnerability. The current version of CVSS is 3.1 and

TABLE I: The vulnerability management tools and supported features

Vulnerability Management Features					
Name	**Scan Network**	**Identification**	**Analysis**	**Evaluation**	**Treatment**
OpenVAS [20]	✓	✓	✓	☐	☐
Rapid7 Nexpose [22]	✓	✓	✓	✓	☐
Vulnerability Manager Plus [24]	✓	✓	✓	☐	☐
Tripwire IP360 [23]	✓	✓	✓	☐	☐
Qualys Vulnerability Management [21]	✓	✓	✓	☐	☐
Skybox Vulnerability Management [25]	☐	✓	✓	✓	☐

TABLE II: Mapping CVSS score to the qualitative rating in some vulnerability databases [2], [29]–[31], [33]

CVSS Score	**Qualitative rating**
0.0	None
0.1-3.9	Low
4.0-6.9	Medium/Moderate
7.0-8.9	High/Important
9.0-10.0	Critical

has three metrics; Base, Temporal and Environmental. The base metric consists of attributes that remains unchanged over time, while the temporal metric contains characteristics that change over time. Environmental metrics are attributes which are unique to the environment. All metrics have a score range from 0 to 10. Out of the metrics, the base metric is considered as the primary score, while the other two are optional in the scoring process.

Table II gives an example of how a CVSS score can be mapped to a qualitative rating used in some VDs. Due to the difference in scoring/ratings among VDs, the same vulnerability can obtain a different score depending on the used VD. Table III provides an example for CVE-2020-8130 in some VDs which describes in section IV-A in details.

IV. Proposed Solution

We think a vulnerability should not analyse in isolation. Therefore, organizational knowledge plays a vital role during the analysis, evaluation and treatment steps in a VRM. The VMTs in Table I provide reports that are based on the severity rating of each detected vulnerability, possible remediation method and reference to the CVE ID. Those reports provide general view of vulnerabilities, and need a domain expert to review and prioritize patching (within the organization). Hence, existing solutions lack the capability to adjust for organizational requirements and automatic patching.

The proposed AC-VRM is aware of the organizational context, and thus provides more relevant (risk) metrics to the organization. Our solution aims to address the complete VRM method in in an automated manner to minimize the time between vulnerability detection and patching. This will be achieved by more efficient vulnerability detection, evaluation, prioritization and automatic patching.

Fig. 2 presents the AC-VRM workflow. The left box labeled *Generic* is related with the process of collection, organization and adaptation of information from multiple VDs. The box labeled *Organization* deals with tasks that are organizational aware, i.e. the self-vulnerability assessment and remediation part of AC-VRM. A short description of AC-VRM workflow follows:

- Collect data: this task collects and updates the information of known vulnerabilities from multiple VDs. This to have as wide knowledge base as possible.
- Pre-process: the information from each VD needs to be adapted into an internal format. Since each VD has its own naming format, we use this step to list the vulnerabilities based on the CVE ID.
- Normalization: this task normalizes the ratings obtained from the used VDs. Depending on VMM setting (basic, standard or restrictive), the normalization from a VD rating to a numeric value can be influenced.
- Vulnerability Management Mode (VMM): is used when translating a VDs qualitative rating to a numeric CVSS score. There are three levels; basic, standard and restrictive. For basic, the lower value in the range is used, standard uses the center value, while restrictive uses the upper range value cf. II.
- Filter: this step is identified the vulnerabilities that effect the organization. The vulnerability will be filtered based on organizational assets.
- Evaluation: vulnerabilities that influences systems in the organization are evaluated according to the organizational knowledge and normalized score (usually done by security experts). The evaluation will generate an initial patch prioritization. AC-VRM uses four priority levels; 0, 1, 2, and 3 which correspond to: ignore vulnerability, immediate patching (within 7 days), patch within 30 days and mitigate with other action.
- Organization knowledge: is the core source of inputs for the workflow. It includes organization system management documents such as information system policies, infrastructure setup, asset inventory and installed software inventory. It could even include reports from VMTs. This information is usually obtained from databases or text files.
- Sort for organization: this task schedule the patching in an order that is as time efficient as possible. It identifies patches for each asset (i.e. infrastructure) and group them together. It could recognize patches that may influence each other, and sequence them in the least disruptive order.
- Patching prioritization: assign due date for each vulnerability in scheduled patching. For example vulnerability

X with priority 1 should be patched in asset Y within 7 days. This step sets a patching deadline for each priority, which is the main criteria in automated patching. It also activates the notification alert for each vulnerability in case the patch deadline passed (e.g. the patching fails or is delayed).

- Automated patching: this task is in charge of patching the vulnerable assets in the organization. The asset under patch will be taken out of production.
- Verification: in this phase, the effected assets are verified to be immune against vulnerabilities (priority 1, and 2). This is done using a verification script or a scanner. If the patch failed to address the vulnerability, the patch priority component is notified and the asset will not be back to production.
- Update: this task in the feedback loop allows to learn the new published vulnerabilities that effects organization. Update step is used to revise the patch priority list when the vulnerability with higher impact are detected or new update of listed vulnerability arrived.

As show in Fig. 2, the major contribution of our solution is additional steps introduced into a VRM, i.e normalization, evaluation, patch priority and automated patching. This paper focuses on the normalization phase as an important prerequisite for comprehensive scoring of vulnerability when using multiple VDs.

A. Normalization

As we mentioned when describing VDs in Sec. III-A, CVE ID is used as a reference by VDs and each VD is updated when new vulnerabilities are published. Furthermore, each VD has its own method to name vulnerabilities, describes the impact and qualitative rating.

To be proactive in protection, we need to collect the updated information from the VDs. This cause some issues, c.f. different CVSS score for the same CVE ID or different scoring method. As an example the CVE-2020-8130, see Table III, scored 8.1 in NVD and medium in USN. The reason for this difference is that USN calculates CVSS qualitative rating of the vulnerability in Ubuntu releases (average of base and environment metrics) while NVD considers CVSS numerical score of the base metrics in its calculation. Therefore, we need to normalize the severity rate of vulnerabilities.

The normalization procedure is provided in Algorithm 1. The *normalizeScore* procedure gets as input the CVE_ID of a vulnerability, the set of VDs used and the value of VMM. Then, for each VD in the set it extracts the severity score for the vulnerability (lines 4 - 7) and finally it returns the average value (line 8). The severity score for a vulnerability given the VMM is computed by the *getSeverityValue* procedure. *getSeverityValue* extracts the severity score for the vulnerability, using *getSecScore*, from VD_i (line 12) and check if it is qualitative or quantitative. In the first case the score is mapped into a numeric value as defined in Table II and, depending on the value of VMM, is taken the lower bound, the center value or the upper bound of the CVSS score range

Algorithm 1 Normalization Algorithm

```
1:  procedure normalizeScore(CVE_ID, VMM, VDset)
2:      /* VDset is the set of VDs considered */
3:      score = 0;
4:      for each VD_i in VDset do
5:          sv = getSeverityValue(CVE_ID, VD_i, VMM);
6:          score += sv;
7:      end for
8:      return ( score = score / VDset.numVDs );
9:  end procedure
10:
11: procedure getSeverityValue(CVE_ID, VD_i, VMM)
12:     ss = getSecScore(CVE_ID, VD_i,)
13:     if isString( ss ) then
14:         switch VMM do
15:             case basic: ss = CVSSmap.lower( ss );
16:             case standard: ss = CVSSmap.center( ss );
17:             case restricted: ss = CVSSmap.upper( ss );
18:     end if
19:     return ( ss ); /* in case ss is numeric it is returned
    directly */
20: end procedure
```

TABLE III: Compare CVSS score of CVE-2020-8130 in different VDs [2], [29], [30], [34]

Database Name	CVSS Score/Qualitative rating
NVD	8.1
USN	Medium
RHSA	Moderate
DSA	9

(lines 14 - 17). The function CVSSmap() is responsible for the mapping and implement also the lower bound, center and upper bound methods. In case the vulnerability score is numeric, it is returned directly without any mapping operated by CVSSmap. For example high qualitative rating corresponds the interval [7.0-8.9] in CVSS score, hence the lower boundary is 7.0 as shows in the Table II.

Table III shows the information retrieves for CVE-2020-8130 in different VDs and Table IV represents the output of normalization step in our solution for the same vulnerability ID.

V. Conclusion

In this paper we presented the AC-VRM method. This augments the current state-of-the-art of VRM, by adding support for organizational knowledge that directs its behavior to generate patching priorities optimal for the organization. AC-VRM operates automatically, thus reducing the discovery/identification-remediation time. Furthermore, as to maximize AC-VRM's coverage, we propose to use input from multiple VDs that is normalized to a VD independent format, while retaining links back to the VD's and the CVE ID. During the normalization process, an organization can choose what level it want to operate via VMM parameter. This gives the method maximum flexibility and adaptability,

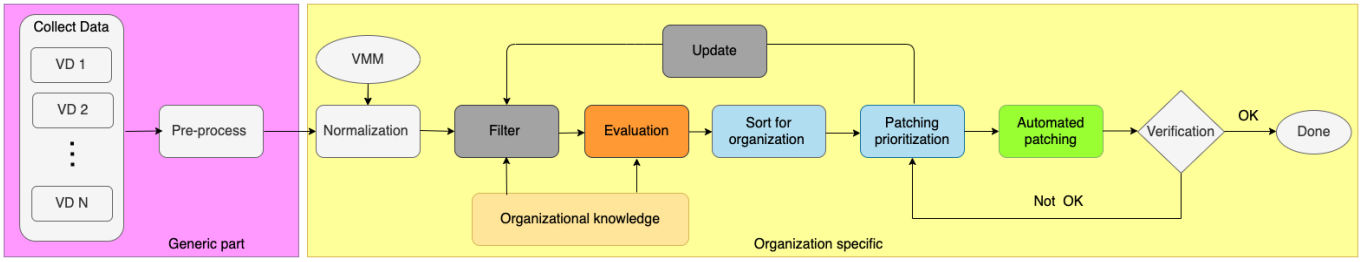

Fig. 2: The AC-VRM workflow. The tasks in the Organization specific part are mapped, using the same colour code, on phases of the self-vulnerability assessment and remediation controller presented in Fig. 1.

TABLE IV: The output of normalization step for given vulnerability ID in basic setting

Vulnerability ID	Normalization Score	Normalization Setting
CVE-2020-8130	6.3	Basic

to find solutions suitable to the organization.

Acknowledgment: The work of E.Casalicchio is partially funded by the SmartDefense project n. RG11916B88C838E8.

References

[1] CIS Controls . http://www.cisecurity.org/controls/. [Online; accessed 5-April-2020].

[2] NIST National Vulnerability Database. https://nvd.nist.gov/. [Online; accessed 8-March-2020].

[3] Debabrata Dey, Atanu Lahiri, and Guoying Zhang. Optimal policies for security patch management. *INFORMS Journal on Computing*, 27(3):462–477, 2015.

[4] V. Katos, S. Rostami, P. Bellonias, N. Davies, A. Kleszcz, S. Faily, A. Spyros, A. Papanikolaou, C. Ilioudis, and K. Rantos. State of vulnerabilities 2018/2019. Technical report, Skybox Security, 2020.

[5] Gary L Guzie. Vulnerability risk assessment. Technical report, ARMY RESEARCH LAB WHITE SANDS MISSILE RANGE NM, 2000.

[6] Eric Yuan, Naeem Esfahani, and Sam Malek. A systematic survey of self-protecting software systems. *ACM Trans. Auton. Adapt. Syst.*, 8(4), January 2014.

[7] Eric Yuan, Sam Malek, Bradley Schmerl, David Garlan, and Jeff Gennari. Architecture-based self-protecting software systems. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 33–42, 2013.

[8] Alberto Huertas Celdrán, Manuel Gil Pérez, Félix J García Clemente, and Gregorio Martínez Pérez. Towards the autonomous provision of self-protection capabilities in 5g networks. *Journal of Ambient Intelligence and Humanized Computing*, 10(12):4707–4720, 2019.

[9] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. Improving vulnerability remediation through better exploit prediction. In *2019 Workshop on the Economics of Information Security*, 2019.

[10] Seungsoo Lee, Jinwoo Kim, Seungwon Woo, Changhoon Yoon, Sandra Scott-Hayward, Vinod Yegneswaran, Phillip Porras, and Seungwon Shin. A comprehensive security assessment framework for software-defined networks. *Computers & Security*, 91:101720, 2020.

[11] R. Vibhandik and A. K. Bose. Vulnerability assessment of web applications - a testing approach. In *2015 Forth International Conference on e-Technologies and Networks for Development (ICeND)*, pages 1–6, 2015.

[12] Y. Tatarinova. Avia: Automatic vulnerability impact assessment on the target system. In *2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP)*, pages 364–368, 2018.

[13] Zhenkai Liang and R. Sekar. Fast and automated generation of attack signatures: A basis for building self-protecting servers. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, page 213–222, New York, NY, USA, 2005. Association for Computing Machinery.

[14] Saad Khan and Simon Parkinson. Review into state of the art of vulnerability assessment using artificial intelligence. In *Guide to Vulnerability Analysis for Computer Networks and Systems*, pages 3–32. Springer, 2018.

[15] Ziwei Ye, Yuanbo Guo, and Ankang Ju. Zero-day vulnerability risk assessment and attack path analysis using security metric. In Xingming Sun, Zhaoqing Pan, and Elisa Bertino, editors, *Artificial Intelligence and Security*, pages 266–278, Cham, 2019. Springer International Publishing.

[16] R. Negi, P. Kumar, S. Ghosh, S. K. Shukla, and A. Gahlot. Vulnerability assessment and mitigation for industrial critical infrastructures with cyber physical test bed. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 145–152, 2019.

[17] Sachin Shetty, Michael McShane, Linfeng Zhang, Jay P Kesan, Charles A Kamhoua, Kevin Kwiat, and Laurent L Njilla. Reducing informational disadvantages to improve cyber risk management. *The Geneva Papers on Risk and Insurance-Issues and Practice*, 43(2):224–238, 2018.

[18] Pankaj Pandey, Anastasija Collen, Niels Nijdam, Marios Anagnostopoulos, Sokratis Katsikas, and Dimitri Konstantas. Towards automated threat-based risk assessment for cyber security in smarthomes, 07 2019. Copyright - Copyright Academic Conferences International Limited Jul 2019; Last updated - 2020-03-04.

[19] Common Vulnerabilities and Exposures(CVE). https://cve.mitre.org/. [Online; accessed 1-June-2020].

[20] Open Vulnerability Assessment Scanner(OpenVAS). https://www.openvas.org/. [Online; accessed 12-May-2020].

[21] Qualys Vulnerability Management. https://community.qualys.com/vulnerability-management/. [Online; accessed 12-May-2020].

[22] Rapid7 Nexpose Vulnerability scanner. https://www.rapid7.com/products/nexpose/. [Online; accessed 12-May-2020].

[23] Tripwire Vulnerability Management. https://www.tripwire.com/products/tripwire-ip360. [Online; accessed 12-May-2020].

[24] Vulnerability Manager Plus. https://www.manageengine.com/vulnerability-management/. [Online; accessed 12-May-2020].

[25] SkyBox Vulnerability Management. https://www.skyboxsecurity.com/vulnerability-management/. [Online; accessed 12-May-2020].

[26] 2020 vulnerability and threat trends. Technical report, SkyBox Security, 2020.

[27] Saad Khan and Simon Parkinson. Towards automated vulnerability assessment. 2017.

[28] James A Kupsch and Barton P Miller. Manual vs. automated vulnerability assessment: A case study. In *First International Workshop on Managing Insider Security Threats (MIST)*, pages 83–97, 2009.

[29] Ubuntu Security Notice. https://usn.ubuntu.com/. [Online; accessed 8-March-2020].

[30] RedHat Security Advisories. https://access.redhat.com/security/security-updates/#/. [Online; accessed 8-March-2020].

[31] Cisco Security Advisories. https://tools.cisco.com/security/center/mpublicationListingDetails.x?docType=CiscoSecurityAdvisory. [Online; accessed 8-March-2020].

[32] Common Vulnerability Scoring System(CVSS). https://www.first.org/cvss/. [Online; accessed 9-March-2020].

[33] First.ORG Inc. Common vulnerability scoring system v3.1: Specification document. 2019.

[34] Debian Security Tracker. https://www.debian.org/security/#DSAS. [Online; accessed 29-April-2020].