Towards Optimal Triage and Mitigation of Context-Sensitive Cyber Vulnerabilities

Soumyadeep Hore, *Graduate Student Member, IEEE*, Fariha Moomtaheen, Ankit Shah, *Member, IEEE*, and Xinming Ou

Abstract—Cyber vulnerabilities are security deficiencies in computer and network systems of organizations, which can be exploited by an adversary to cause significant damage. The technology and security personnel resources currently available in organizations to mitigate the vulnerabilities are highly inadequate. As a result, systems routinely remain unpatched, thus making them vulnerable to security breaches from the adversaries. The potential consequences of an exploited vulnerability depend upon the context as well as the severity of the vulnerability, which may differ among networks and organizations. Furthermore, security personnel tend to have varying levels of expertise and technical proficiencies associated with different computer and network devices. There exists a critical need to develop a resource-constrained approach for effectively identifying and mitigating important context-sensitive cyber vulnerabilities. In this article, we develop an advanced analytics and optimization framework to address this need and compare our approach with rule-based methods employed in real-world cybersecurity operations centers, as well as a vulnerability prioritization method from recent literature. First, we propose a machine learning-based vulnerability priority scoring system (VPSS) to calculate the priority scores for each of the vulnerabilities found in an organization's network and quantify organizational context-based vulnerability exposure. Next, we propose a decision-support system, which consists of a two-step sequential optimization approach. The first model selects the high priority vulnerability instances from the dense report subject to resource constraints, and the second model then optimally allocates them to the security personnel with matching skill types for mitigation. Experiment results conducted using a real-world vulnerability data set show that our approach 1) outperforms both the rule-based methods and the vulnerability prioritization method from literature in prioritizing context-sensitive vulnerabilities, which are found across highly susceptible organizationally relevant host machines, and 2) maximizes the pairs of vulnerability instance type and the respective security analyst skill type for optimal mitigation.

Index Terms—Context-sensitive vulnerability triage and mitigation, cyber vulnerability management, machine learning, mixed integer programming, sequential optimization, vulnerability priority scoring system

1 Introduction

Malicious actors are actively looking to exploit any weaknesses found in the computational logic of software and hardware components in an organization's network. Upon identification, these weaknesses, also referred to as vulnerabilities, are reported in the National Vulnerability Database (NVD), which is sponsored by the Department of Homeland Security's National Cyber Security Division. The number of new vulnerabilities reported in the NVD has increased annually in the last three years, and these vulnerabilities are now more than two times what they were in 2016 [1]. Concurrently, the number of new vulnerabilities has significantly increased in organizations' networks. However, the technology and security personnel resources have lagged behind in adequately matching the effort needed to mitigate them, which has resulted in an asymmetric advantage for these malicious actors.

Fig. 1 shows the typical vulnerability management process employed by many organizations. In this process, the software and hardware components of an organization's network are scanned periodically to find vulnerabilities that may have been reported in the NVD. There are many types of vulnerability scanners available in the market. Some of the popular vendors include Tenable, Qualys, and IBM. The vulnerability report contains valuable information about the identified instances of vulnerability, such as the common vulnerability exposure (CVE) code, host name, description, and the common vulnerability scoring system (CVSS) value indicating the severity rating, among others. There are many available actions to mitigate each vulnerability found in this dense report. These actions include upgrading of the software, disabling/disconnecting the service, applying a vendor-supplied patch, and adding an IP filter, among others. Security analysts have varying skill sets associated with different computer and network devices. For instance, some analysts are more proficient in securing web server infrastructure compared to mobile user environment. Some analysts are more proficient with Microsoft systems compared to Unix-based systems. Based on their skill sets and availability, security personnel take appropriate actions to mitigate the threats posed by the vulnerabilities. The two commonly employed vulnerability triage and mitigation strategies are 1) a CVSS value-based plan and 2) an analysts' key performance indicator (KPI)-driven plan. In the former strategy, the vulnerability

• *The authors are with the University of South Florida, Tampa, FL 33620 USA. E-mail: {soumyadeep, fmoomtaheen, ankitshah, xou}@usf.edu.*

Manuscript received 15 April 2021; revised 27 January 2022; accepted 8 February 2022. Date of publication 22 February 2022; date of current version 14 March 2023.
(Corresponding author: Ankit Shah.)
Digital Object Identifier no. 10.1109/TDSC.2022.3152164

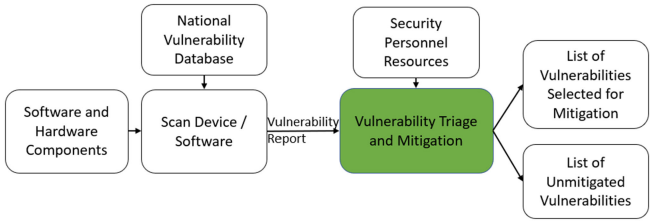

Fig. 1. Typical vulnerability management process.

instances selected for mitigation are those that maximize the cumulative CVSS value by taking into account their severity scores. In the latter strategy, security analysts maximize the remediation of vulnerabilities by selecting the ones that are easier (and that take less time) to mitigate.

The above myopic approaches yield sub-optimal outcomes because of the following three major problems. 1) Vulnerability instances with a lower severity continue to persist in the network for a long time and may result in serious consequences; for instance, the infamous WannaCry cyberattack on machines with an unpatched Microsoft Windows operating system cost the world economy billions of dollars. 2) A CVSS base score does not reflect the true severity of a vulnerability, as it does not consider organization-specific context-based factors such as the relevance of the segment (location) of the network where the vulnerability is reported, the existing level of protection in that respective segment of the network, and the presence of high-value assets in that segment. The potential consequences stemming from an exploited vulnerability with a given CVSS value may differ from one network to another and also from one organization to another. 3) The sub-optimal approach to the allocation of security personnel without considering their expertise and skills for vulnerability mitigation may also directly impact the security posture of an organization.

In this paper, we address the aforementioned challenges and the critical need to improve the security posture of an organization by optimizing the vulnerability triage and mitigation process. The contributions of this research are as follows. First, we developed a novel machine learning-based vulnerability priority scoring system (VPSS), which takes into account organizational context (through qualitative factors) along with the assigned CVSS value from the NVD for each vulnerability found in the network. We incorporate the domain expertise of humans for feature engineering to develop the machine learning models for the VPSS. In this research study, we worked with security analysts at a cybersecurity operations center (CSOC) to assign quantitative values to the qualitative factors identified for the calculation of the VPSS scores for the vulnerabilities. We used a subset of the vulnerability instances found in a real-world CSOC and their respective validated VPSS scores from the security personnel as a training data set to estimate the VPSS scores of all the vulnerability instances found in the scan reports. The problem of selecting important context-sensitive vulnerabilities from this dense report and allocating them to the limited number of skilled security personnel is not trivial. Hence, second, we developed a decision-support system consisting of a two-step sequential optimization approach that optimizes the security posture of an organization by first selecting the high priority vulnerabilities for mitigation based on their VPSS scores in a resource-constrained environment, and then allocating them to the available security personnel with matching skill sets. This decision-support system takes the dense vulnerability scan report with the VPSS-generated vulnerability scores, along with the security personnel availability and skill sets, as inputs. The output of the system is the mitigation action plan for the organization, which consists of the selected vulnerabilities and their respective allocations to the security personnel for mitigation. Our proposed framework is scalable and provides organization-specific customizations that can assist any organization with prioritizing and mitigating context-sensitive vulnerabilities. Third, we provided insights obtained using our proposed approach by comparing our results with the other commonly employed rule-based methods and a vulnerability prioritization method [2] from recent literature. Our experiment results involving real-world vulnerability scan data show that our approach outperforms these methods in prioritizing context-sensitive vulnerabilities with respect to high-value assets (such as important sub-domains), relevance (web and database servers), and highly susceptible host machines (with a lower level of protection in the network). In addition, our proposed approach is able to maximize the pairing of vulnerability instance type and the respective security analyst skill type for mitigation, thereby resulting in an efficient vulnerability management for the organization.

The paper is organized as follows. Section 2 presents the related literature. Section 3 presents the proposed framework, which consists of the machine learning-based vulnerability priority scoring system and the two-step sequential optimization models. Also, model parameters, mathematical formulation for the optimization models, computational complexity, algorithm, and baseline rule-based methods are discussed here. Section 4 presents the numerical experiments performed using real-world vulnerability scan data. Section 5 presents the experiment results and comparisons with other methods. Lastly, Section 6 discusses the insights obtained from this research study and provides conclusions.

2 Related Literature

A 2018 report by the Council of Economic Advisors, an agency within the Executive Office of the President, estimated that malicious cyber activity cost the U.S. economy between $57 billion and $109 billion in 2016 [3]. This number is expected to increase until organizations deploy adequate solutions to protect themselves from such malicious cyber threats. Due to this significant increase in the number of vulnerabilities found in a network and the lack of available resources, organizations must triage vulnerabilities for mitigation. The common vulnerability scoring system (CVSS) is a widely used mechanism to triage vulnerabilities. Several changes have been made in the architecture of the CVSS score over the years. The most recent CVSS version is 3.1, which consists of eight base metrics, three temporal metrics, and four environmental metrics [4]. The base metrics are constant over time and represent the inherent characteristics of vulnerabilities. The base metrics can be further decomposed into two different groups: exploitability

metrics and impact metrics. The exploitability metrics represent the ease and technical means required to exploit the vulnerability, whereas the impact metrics represent the potential impact of a successful exploitation. Temporal metrics reflect the characteristic of vulnerability that may change over time and the environmental metrics represent the characteristic of a vulnerability that are environment specific [4]. Mell *et al.* [5] comprehensively analyzed the CVSS metrics when version 2.0 was the state-of-the-art. Anecdotal evidence indicates that most organizations have since used CVSS base scores to prioritize the vulnerabilities for mitigation. The environmental and temporal metrics of the CVSS scores are omitted as NVD employs CVSS base score metric to be the severity indicator of all recorded vulnerabilities [6]. Although the CVSS score consists of various environmental metrics, their computation is complicated and not well proven [7]. The omitted environmental and temporal metrics account for an organization's context. Due to the missing contextual information, the NVD scoring is of limited use [8]. Researchers and practitioners argue that this score alone is not a good indicator of the compromise time of a device/network, or the potential consequences that may arise from exploitation of a vulnerability [2], [6], [9], [10]. Some research studies have tried to bridge this gap. For instance, Cavusoglu *et al.* [11] presented a game theoretic approach that facilitates better understanding of vendor-CSOC relationship by incorporating cost-benefit analysis, and Allodi and Massacci [12] incorporated external information such as black-market exploit data to obtain a more statistically significant indication of the true severity of a vulnerability. There remains, however, an imbalance between the total available time and the total time required to mitigate all vulnerabilities found in an organization's network.

Farris *et al.* [2] proposed two performance metrics, namely: total vulnerability exposure and time to vulnerability remediation, and then optimized the selection of vulnerabilities for mitigation by developing a multi-objective mixed integer programming optimization model. Shah *et al.* [13] investigated two approaches: individual attribute value optimization and multi-attribute value optimization for vulnerability selection. The results indicated that using a multi-attribute value optimization is superior to optimizing the selection of vulnerabilities with respect to a single attribute. Similar multi-objective optimization models have been developed for other fields, which include implementation of intersection signal control with the goal of minimizing emission and travel delay [14] and evaluation of the transit signal priority using an integrated traffic signal control consisting of a genetic algorithm and artificial neural networks [15], among others.

Optimization approaches are used for the allocation of human resources to improve products and services in various fields. For example, a goal programming model is developed for strategic planning and allocation for limited human resources in a healthcare organization [16]. Similarly, an optimization system is developed for selecting profitable projects from a set of possible alternatives while also performing the optimal allocation of human resources for those selected projects [17]. Based on the varying degree of difficulties, a multi-objective, multi-factorial optimization model is developed to decompose and dynamically allocate resources [18]. A methodology based on dynamic programming is presented to assign human resources to software development projects in [19]. An inverse optimization model for human resource allocation is developed by taking into consideration the competency disadvantages of individuals [20]. A conceptual framework for software resource allocation and selection is developed to maximize a company's profit based on the levels of software skills [21]. A regression test prioritization technique is developed to cover maximum faults in minimum time through an ant colony optimization [22].

Our research differs from the aforementioned studies by proposing a unified framework that 1) prioritizes vulnerabilities for mitigation by developing a novel vulnerability priority scoring system that takes into account both the organizational context and the CVSS-based severity ratings, and 2) optimizes the selection and allocation of the vulnerability instances to security personnel by matching their skill sets with the types of vulnerabilities. To the best of our knowledge, this research study is the first one to consider the attributes of security personnel in making vulnerability allocation decisions for optimizing the security posture of an organization. This problem is non-trivial due to the complexity of the computer networks, the vast amount of vulnerabilities that exist on the host machines/devices, and the limited number of resources available for vulnerability management. Efficient vulnerability management can significantly streamline security operations and improve the overall security posture of an organization.

3 Advanced Analytics and Optimization Framework for Context-Sensitive Vulnerability Management

There is a critical need to improve the security posture of an organization by optimizing the vulnerability triage and mitigation process, which can be achieved by adding organization-specific information to prioritize the mitigation of context-sensitive vulnerability instances and assigning them to appropriate security analysts with matching skill sets. Fig. 2 shows a schematic of the proposed framework for vulnerability management. The software and hardware components of an organization's network are first scanned to find the vulnerabilities reported in the NVD. The vulnerability report is then provided as an input to the machine learning-based vulnerability priority scoring system (VPSS) that assigns scores to the vulnerability instances. The updated vulnerability report with the VPSS scores is then provided as an input to the decision-support system, which selects the vulnerability instances for mitigation and allocates them to the appropriate security personnel. The proposed framework comprises the following key components: 1) a novel machine learning-based VPSS, which takes into account the organization-specific context in which vulnerabilities are reported, among other factors, and 2) a decision-support system with a two-step sequential optimization approach that take into account the large vulnerability scan data, their VPSS scores, historical vulnerability mitigation data, and the security personnel information (availability and skill sets) to obtain an optimal mitigation plan. First, we describe each of

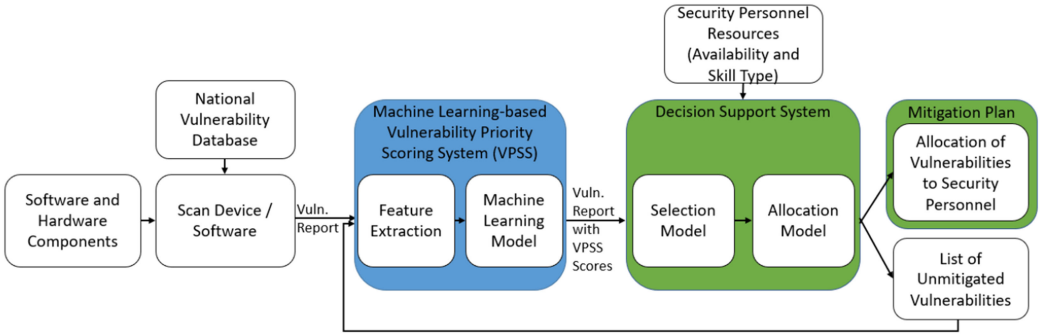

Fig. 2. A schematic of the proposed framework for context-sensitive vulnerability triage and mitigation.

the aforementioned components, followed by the methods used for comparison.

3.1 Vulnerability Priority Scoring System (VPSS)

We present a novel vulnerability priority scoring system (VPSS) to calculate the priority scores for each of the vulnerabilities found in an organization's network. The VPSS takes into account organization-specific context-based information to determine the vulnerability priority scores. The factors considered in the VPSS are as follows: a) the relevance of the segment (location) of the network where the vulnerability is reported, b) the level of protection provided in the respective segment, c) the presence of high-value assets in that segment, and d) the CVSS value of the vulnerability. An affinity diagram of the factors is shown in Fig. 3. By using a quantitative value function hierarchy process [23], a preference in the form of a weight, w_i, is assigned to each of these factors (indexed by i). These weights are obtained from the stakeholders (security team) at the organization to initialize the VPSS model. These initial weights can be later adjusted based on the security personnel feedback received during triage and mitigation.

A list of responses is created for each of these factors, based on our conversations with the security experts. As shown in Table 1, there are at least three categories of responses available for the first three factors (high, medium, and low). Based on the organizational needs, this list can be expanded to add more qualitative responses (such as critical, high, medium, and low). Through an elicitation process (with the help of surveys and interviews), the responses to each of these factors are obtained from the stakeholders, which are then transformed into numerical values. To keep the scale of each of the factors in the same range, the numerical values are normalized between 0 and 1. However, to have each factor's contribution to the calculation of the priority score of a vulnerability instance, the lowest transformed numerical value is clipped at 0.1. The categorical responses for each factor are assigned numerical values as follows. The categorical response with the highest priority is assigned a value of 1 and that of the lowest priority is assigned a value of 0.1. The responses that are in between the highest and the lowest priorities are assigned values that are uniformly spaced between them. A similar approach to assigning numerical values to qualitative factors is used in [24] to compute the risk associated with an intrusion detection system alert. For instance, a high rating for the relevance (say, factor $i = 1$) of a vulnerability instance j will be given a value of $u_{i=1,j} = 1$, whereas a low (medium) rating will assign $u_{i,j}$ to 0.1 (0.5). Similarly, the values of the CVSS score factor are normalized by taking into consideration the largest value that the factor can take (10 for CVSS score) with the minimum value of 0.1. The purpose of such a scheme of assigning numerical values is to maintain hierarchy among the categorical responses for each factor and obtain a priority score using an additive value function. Finally, each vulnerability instance, j, is assigned a VPSS score, $V_j = \sum_i (w_i * u_{i,j})$. We define the vulnerability exposure score of an organization as $C = \sum_j V_j$, for all j unmitigated vulnerabilities. Note that this model can be tailored to the specific cybersecurity operation requirements by adding more factors.

The vulnerability scan reports tend to be very dense for large organizations. Hence, to make this scoring mechanism scalable and implementable, we propose the use of an advanced analytics approach by using a supervised machine learning model to estimate the VPSS scores. Fig. 4 shows a schematic of the VPSS scoring process. This is the process followed to train the machine learning model. It is to be noted that this process is repeated on a periodic basis to capture changes in the network or new types of vulnerabilities and also to adjust the priorities (factor weights) of the organization. As explained earlier in this section, first,

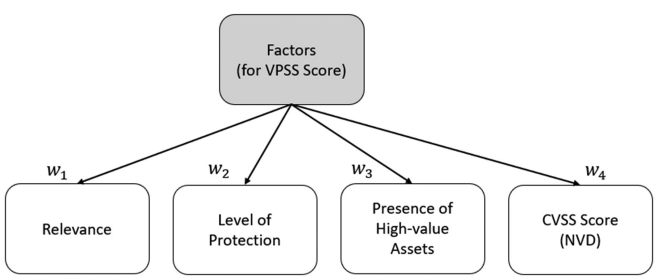

Fig. 3. Factors in the vulnerability priority scoring system (VPSS).

TABLE 1
Responses for the Factors

Factor	Response
Relevance	High, Medium, or Low
Level of Protection	High, Medium, or Low
Level of Asset Criticality	Crtitical, High, Medium, or Low
CVSS Score	Numerical

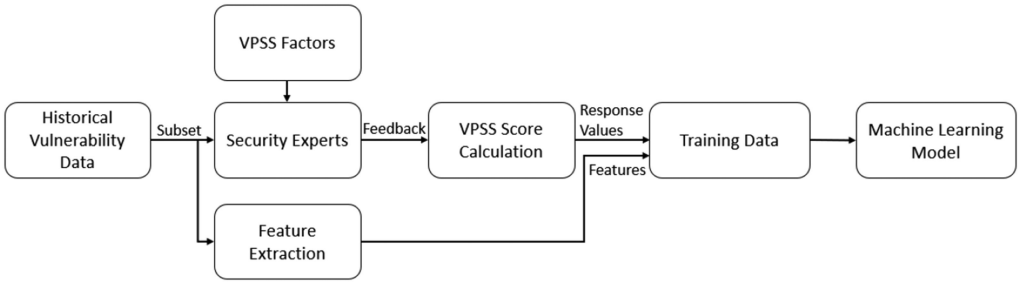

Fig. 4. A schematic of machine learning-based VPSS scoring process.

the security team in an organization assists in the priority scoring of the vulnerability instances by identifying the preferences (weights) for the factors and assigning numerical values to the qualitative responses, with their organizational knowledge and expertise. Next, this subset of vulnerabilities is used as a training data set to build and train a machine learning model. The VPSS scores in this training data set are considered to be the ground truth response values and the mean squared error (MSE) is used as a performance metric to train various types of models (such as linear, tree-based, and nonlinear). This vulnerability data set with all the attribute information is in a natural language format. Hence, natural language processing-based feature extraction techniques are utilized to obtain the important features and convert them into numerical values. These features are then passed on to the machine learning model with the respective VPSS scores as the response values for the vulnerability instances. The model with the lowest MSE value obtained using the training samples is then chosen as the machine learning model to be used in the proposed framework (as shown in Fig. 2). Next, we present the decision-support system that selects context-sensitive vulnerabilities and allocates them to the appropriate security personnel for mitigation. The output of the allocation model is the allocation of the selected vulnerability instances from the selection model to the security personnel for mitigation.

3.2 Decision-Support System

The decision-support system consists of two optimization models, which are executed in a sequential manner: 1) the selection model and 2) the allocation model. The objective of the selection model is to select vulnerability instances for mitigation that minimizes the vulnerability exposure score subject to the total time available for mitigation in a given time-period. The objective of the allocation model is to allocate the selected vulnerability instances obtained from the selection model to the security personnel for mitigation by maximizing the matching pairs of the type of vulnerability instance and the skill type of the analyst. However, it is to be noted that having all skill types of security personnel in all time-periods is not attainable in real-world CSOCs due to staffing and scheduling issues. In a real-world scenario, a sub-optimal matching of the vulnerability type and the analyst skill type will impact the time taken to mitigate the vulnerability instance due to the lack of familiarity and expertise of the security personnel. In the cases where all matching pairs are not possible, additional time will be required (i.e., $T + \delta$) to mitigate the selected vulnerabilities. It is to be noted that the objective of this research study is to minimize the context-sensitive vulnerability exposure of an organization, which cannot be achieved by combining the constraints from the aforementioned models into one model as that could restrict the selection of certain high priority vulnerability instances where matching analyst skill types are unavailable. Hence, by decomposing the solution approach into two separate but sequential models, we make the decision-support system applicable in a real-world security team environment, where all types of skill levels may not be attainable for all time-periods. Next, we present the mathematical formulation for the two optimization models. The notations used in the mathematical formulations are defined in Table 2.

3.2.1 Mathematical Formulation for Selection Model

Below we present the input parameters, decision variables, objective function, constraints, and the outputs of the selection model.

Input parameters:

- The VPSS scores for all vulnerability instances, $V_j \forall j$.
- Expected time taken to mitigate a vulnerability instance j, S_j.
- Total number of vulnerability instances in the scan report, J.
- Total personnel-hours available for a given time-period, T.

Decision variables:

- $z_j = 1$ if vulnerability instance j is selected, and 0 otherwise.

Objective function: The objective of the selection model is to select vulnerability instances for mitigation that minimizes the vulnerability exposure score (i.e., select vulnerability instances for mitigation that maximizes the cumulative VPSS score) subject to the total time available for mitigation in a given time-period.

$$y = Max \sum_{j=1}^{J} V_j * z_j \tag{1}$$

Constraint: The constraint for the selection model is as follows:

- The total time taken to mitigate the selected vulnerability instances must not be higher than the total personnel-hours available for the given time-period, which is expressed as:

TABLE 2
Definitions of Notations

Notation	Definition
Indices:	
j	Vulnerability instance index.
k	Analyst skill type index.
Inputs:	
J	Total number of vulnerability instances in the scan report.
K	Total number of analyst skill types.
T	Total personnel-hours available in a given time-period.
R_k	Total number of hours available for analyst skill type k.
S_j	Expected mitigation time for j.
$H_{j,k}$	Expected mitigation time for j by k (when types differ).
V_j	VPSS score for vulnerability instance j.
Variables:	
z_j (Binary)	Selection of vulnerability instance j.
$a_{j,k}$(Binary)	Allocation of vulnerability instance j to skill type k.
d_k	Amount of overtime (in hours) for analyst skill type k.
m	Maximum overtime (in hours).

$$\sum_{j=1}^{J} S_j * z_j \leq T \tag{2}$$

Output: The output of the selection model is the set of vulnerability instances selected for mitigation. Next, we present the formulation for the allocation model.

3.2.2 Mathematical Formulation for Allocation Model

The allocation model assigns the selected vulnerabilities to the security personnel for mitigation by taking into account the vulnerability type and the skill type of the security analysts. To account for real-world conditions in a resource-constrained CSOC environment, where enough numbers of the required analyst skill types are not available during each iteration (say, on a weekly-basis), in this formulation, we consider an additional time ($x\%$) that would be required by an analyst to mitigate a vulnerability instance in the sub-optimal pairs. This is an algorithmic technique, in which a penalty in the form of an additional time is imposed on sub-optimal pairings of the vulnerability instances and security personnel to obtain maximum number of matching pairs. Furthermore, we add the available personnel hours of all the analysts with the same skill type and use this information as an input for the vulnerability allocation decision-making. We present a novel mathematical formulation for this allocation model by posing the problem as a minimization of the maximum overtime that is created for any analyst skill type. A unique advantage of this objective function is that this model will balance the overtime, as uniformly as possible, among all the skill types of the analysts. Below we present the input parameters, decision variables, objective function, constraints, and the outputs of the allocation model.

Input parameters:

- Total number of hours available for analyst skill type k, R_k.
- Set of vulnerability instances selected by the selection model, L, where $|L| = \sum_{j=1}^{J} z_j$
- Expected time taken to mitigate the vulnerability instance j by security personnel with skill type k, $H_{j,k}$. It is to be noted that $H_{j,k} = S_j$, where the type of vulnerability instance matches the skill type of the analyst, and $H_{j,k} = S_j + x\%$, otherwise.

Decision variables:

- $a_{j,k} = 1$ if vulnerability instance j is allocated to analyst skill type k, and 0 otherwise.
- Amount of overtime (in hours) created for analyst skill type k, d_k.
- Maximum overtime (in hours), m

Objective function: The objective of the allocation model is to maximize the matching pairs of the type of vulnerability instance and the skill type of the analyst by minimizing the maximum overtime that is created for any analyst skill type.

$$q = Minm \tag{3}$$

Constraints: The constraints for the allocation model are as follows.

- The total time requirement for analyst skill type k to mitigate the allocated vulnerability instances is expressed as

$$\sum_{j=1}^{|L|} H_{j,k} * a_{j,k} \leq R_k + d_k \forall k \tag{4}$$

- Ensuring allocation of all the selected vulnerability instances to security personnel is given by

$$\sum_{k} a_{j,k} = 1 \forall j \in L \tag{5}$$

- The maximum overtime among the analyst skill types is calculated by:

$$d_k \leq m \forall k \tag{6}$$

Output: The output of the allocation model is the allocation of the selected vulnerability instances from the selection model to the security personnel for mitigation. Next, we present the algorithm for the decision-support system.

3.2.3 Algorithm and Computational Complexity

Algorithm 1 provides the implementable steps for the sequential optimization models in the decision-support system.

Algorithm 1: Sequential Optimization Algorithm for Vulnerability Management.

Input: VPSS scores for all vulnerability instances, $V_j \forall j$; expected time taken to mitigate a vulnerability instance j, S_j; additional time for mitigation for sub-optimal pairs, x%; expected time taken to mitigate the vulnerability instance j by security personnel with skill type k, $H_{j,k}$ ($H_{j,k} = S_j + x\%$); total number of vulnerability instances in the scan report, J; total personnel-hours available for a given time-period, T; total number of hours available for analyst skill type k, R_k.

Output: Selected vulnerabilities for mitigation and their allocation to security personnel, $a_{j,k} \forall j, k$.

/* Initiate a solution search using an integer programming solver */ **repeat**

for *a set of* $z_j = 1$, *check for feasibility:* **do**

$\sum_{j=1}^{J} S_j * z_j \leq T$ /* Available time constraint */

end

if *Feasible* **then**

$y = Max \sum_{j=1}^{J} V_j * z_j$ /* Max cumulative VPSS score */

end

until *Stopping criteria /* Optimal value for y is found */*

/* Initiate a solution search using an integer programming solver */ **repeat**

for *a set of* $a_{j,k} = 1$, *check for feasibility:* **do**

$\sum_{j=1}^{|L|} H_{j,k} * a_{j,k} \leq R_k + d_k \forall k$ /* Time reqd./skill type */ $\sum_k a_{j,k} = 1 \forall j \in L$ /* All instances are allocated */ $d_k \leq m \forall k$ /* Maximum overtime */

end

if *Feasible* **then**

$q = Minm$ /* Min max overtime (max matching pairs) */

end

until *Stopping criteria /* Optimal value for q is found */*

return $a_{j,k} \forall j, k$.

The research problem of vulnerability instance selection and allocation to security personnel for mitigation is formulated as a sequential mixed integer programming problem. Integer programming problems belong to the NP complexity class [25] [26]. The complexity of the research problem is 2^{J*A}, where J is the total number of vulnerability instances in the scan report and A is the total number of security analysts in the organization. To reduce the complexity of the research problem we consider the allocation of vulnerability instances to the different skill types of analysts that are available in an organization. $K < A$ represents the total number of analyst skill types.The complexity is further reduced by decomposing the problem into two parts and then solving them separately. The complexity of the selection model is 2^J and the allocation model is $2^{|L|*K}$, where L is the set of vulnerability instances selected for mitigation by the selection model and $|L| << J$. The computational time for the instances of the problem solved using the proposed two-step optimization approach on the real-world vulnerability scan data was under 1 minute on a 32GB RAM 8-core processor machine using Gurobi solver. We compare the performance of our approach against other methods, which are described next.

3.3 Other Methods for Performance Comparison

Our proposed framework is compared against the rule-based methods that are currently employed by the CSOCs, namely, the CVSS value-based and the key performance indicator (KPI)-driven methods, and a recently proposed vulnerability prioritization method, VULCON, from the literature.

The CVSS value-based method takes into account the CVSS values (base scores) of the vulnerability instances and selects the ones for mitigation that maximize the cumulative CVSS value in a resource-constrained environment. The objective of this method is to select vulnerability instances for mitigation that maximize the cumulative CVSS value in the given time-period. The objective function is expressed as

$$Max \sum_{j=1}^{J} D_j * z_j, \tag{7}$$

where $D_j \forall j$ represents the CVSS values for all vulnerability instances in the scan report. The resource constraint in this method is the same as in Equation (2).

The KPI-driven method takes into account the expected mitigation time of the vulnerability instances in order to maximize the number of vulnerability instances selected for mitigation. The objective of this method is to maximize the total number of vulnerability instances selected for mitigation in the given time-period. The objective function is expressed as

$$Max \sum_{j=1}^{J} z_j \tag{8}$$

The resource constraint in this method is the same as in Equation (2).

The recently proposed VULCON strategy from literature prioritizes the selection of vulnerabilities with respect to the CVSS severity score, persistence, and the chronological age of the vulnerabilities found in the network. This strategy also takes into account mission criticality when triaging the vulnerabilities for selection. The algorithm for this strategy is given in [2].

The output obtained from the above methods is the set of vulnerability instances selected for mitigation. In the next two sections, we describe the numerical experiments that are conducted and compare the results obtained by using our proposed approach with those of the aforementioned methods.

4 Numerical Experiments

We worked very closely with a CSOC and conducted our experiments using real-world vulnerability scan data. The use of real-world data and an active connection with the CSOC adds a degree of pragmatism to this research at a time when computer security research is disproportionately based on synthetically generated data [27]. The real-world data sets contained detailed information about the vulnerability instances and the respective host machines found in

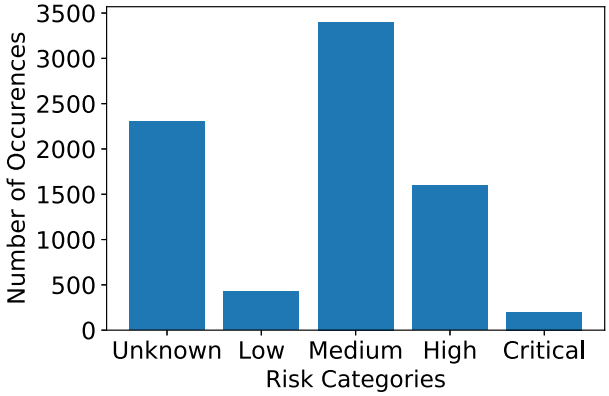

Fig. 5. Number of vulnerability instances per CVSS Severity rating.

the organization's network. This information along with the organizational VPSS factor preferences and the qualitative (categorical) responses for each of them were obtained from the security personnel. These qualitative responses were then assigned numerical values based on certain rules. Finally, the VPSS scores were calculated for the vulnerability instances with the aforementioned information. These scores were then used as the ground truth response values for training the ML model (as shown in Fig. 4). The skill type and expertise of security analysts is given as an input by the CSOC to the decision-support system (in particular, to the allocation model). This information can also be obtained from the historical vulnerability mitigation data and adjusted/changed by the CSOC manager, where required. Based on the various analyst skill types and expertise, we categorized the types of vulnerabilities in the data sets, and used this information to match the analyst skill types to the vulnerability instances. From our conversations with the CSOCs, we learned that it is not a common practice to log the exact time taken to mitigate a vulnerability instance, however, it could be estimated. Hence, keywords from the solution attribute field of the vulnerability instances found in the data sets were assigned an approximate mitigation time (a similar approach is used in [2]). Next, we provide a detailed description of the data sets used in the experiments and the experimental methodology.

4.1 Data Description

Our data sets were obtained using the following two applications: Tenable's Nessus vulnerability scanner and Lansweeper. The Nessus vulnerability scanner produces a report containing all known vulnerability instances that are present in an organization's network. This report contains information about the vulnerability and the host machine, which includes the common vulnerability exposure (CVE) code, host name, description, and the common vulnerability scoring system (CVSS) value indicating the severity rating, among other currently known factors. The Lansweeper report contains information pertaining to the types and versions of software on each machine in the computer network. We conducted our experiments using different data sets. In this section, we explain our largest test data set, which contained 8031 vulnerability instances found on 184 unique hosts. Fig. 5 shows the number of vulnerability instances for each of the CVSS severity ratings in this data set. It can be seen that there is a greater number of vulnerability instances belonging to the medium severity rating, when compared to the critical, high, and low severity ratings in this data set, which is typical in an organization's network. The majority of the vulnerability instances were found to be associated with ports 80 (2000), 3389 (1251), and 443 (1128).

Through our conversations with the security personnel at the CSOC and recent literature survey [2], we estimated the personnel hours required for mitigating each of the vulnerability instances in the data set. We created four bins (groups) based on certain keywords associated with the solution attribute of the vulnerability instances in the data set. Each bin was given an estimated time range for vulnerability mitigation. Based on the keyword found in the solution attribute of a vulnerability instance, a mitigation time estimate was randomly selected from the time range of the respective bin. Table 3 shows the four bins, the estimated time range, and a sample of the keywords used to create these bins. For example, a vulnerability instance with the keyword 'purchase certificate' would be assigned a random time between 0.5 and 1 h for its mitigation. Next, we explain the experimental methodology.

4.2 Experimental Methodology

In the experimental methodology, we describe the steps involved in conducting the experiments. We first describe the experimental methodology for the machine learning-based VPSS scoring model, followed by the decision support system for selection and allocation of vulnerability instances for mitigation.

4.2.1 *Machine Learning-Based VPSS Scoring Model*

As shown in Fig. 4, first, a representative set of vulnerability instances is selected for the VPSS score calculation. We selected vulnerability instance samples (90,000) from historical data sets (reports) to create a training data set for building the machine learning (ML) model(s). We worked with

TABLE 3
Personnel-Hours Estimation

Bins	Estimated Time Range	Important Keywords
Bin 1	0.5 to 1 Hour	'purchase certificate,' 're-issued certificate,' 'disable sslv3,' 'disable terminal services,' 'contact vendors,' 'purchase ssl certificate'.
Bin 2	1 to 3 Hours	'reconfigure application ciphers,' 'restrict ips database,' 'reboot technet,' 'limit traffic port'.
Bin 3	3 to 6 Hours	'patch windows,' 'upgrade flexnet,' 'version update,' 'install patches,' 'upgrade PHP version,' 'server upgrade'.
Bin 4	6 to 9 Hours	'update windows version,' 'system update,' 'system upgrade'.

TABLE 4
Source and Method for Obtaining Factor Responses

Factor	Source	Method
Relevance	Host Machine Type	Lansweeper, Survey
Level of Protection	Software Version	Lansweeper, Nessus Scanner, Survey
Level of Asset Criticality	Sub-domain Importance	Survey
CVSS Score	NVD	Nessus Scanner

TABLE 6
Sample Scores for the Level of Protection Factor in the VPSS

Operating System	SQL Version	Response (Score)
Windows OS 2008 or older	2012 or older	High (1)
Windows OS 2016 - OS 2012	2016 or older upto 2012	Med. (0.5)
Windows OS 2016 or newer	2016 or newer	Low (0.1)

the security personnel at the CSOC to identify their preference for the various VPSS factors (as shown in Fig. 3). Through a combination of information from reports, questionnaire surveys, and discussions, the qualitative (categorical) responses for the factors were obtained. Table 4 shows the factors, sources, and methods used to elicit the responses, and Table 5 shows a sample list of questions/information needed. The numerical values were then assigned to these responses, as described in Section 3.1. The definitions of the factors and the rule-based method of assigning the qualitative response and its corresponding numerical value for each of the factors are described below.

- The relevance of the vulnerability instance was quantified based on the type of host machine: a web server, a database server, or an important stakeholder machine. If a vulnerability was found on a web or a database server (i.e., high relevance), then the factor was assigned a score of 1; else if the host machine was identified to be that of an important stakeholder in the organization (i.e., medium relevance), then it was assigned a score of 0.5; else a score of 0.1 was assigned (indicating low relevance).
- The level of protection factor was quantified using the version of the software (i.e., the version of the operating system (OS) or special systems such as the SQL) running on the host machine. The older the software version, the less protection it may provide against new cyber threats due to the limited (or nonexistent) support provided by the vendor. If the software version of the host machine was not supported by the vendor, then the level of protection was deemed low and hence the qualitative response was high (i.e., the priority was high). Table 6 shows a sample of the score assignment for the level of protection factor in the VPSS. The information about the software version was extracted using Lansweeper. For instance, an operating system older than Microsoft Windows 2008 or an SQL server version of 2008 or older was considered to have the lowest level of protection and a vulnerability instance on such a machine was assigned a score of 1 for this factor.
- There were four levels considered for the level of asset criticality factor: critical-value, high-value, medium-value, and low-value. These were identified by the security personnel at the CSOC and they represented four different sub-domains with the respective level of asset criticality associated with them. A factor score of 1, 0.5, 0.25, or 0.1 was assigned to each vulnerability instance, based on the sub-domain of the host machine. For instance, a vulnerability instance found on a machine in the critical-value sub-domain was assigned a score of 1 for the respective factor.
- The CVSS scores were extracted in numerical form from the vulnerability scan report and were normalized between 0.1 and 1, and accordingly scores were assigned for this factor.

Finally, a composite VPSS score was calculated by taking the weighted sum of the scores of all the aforementioned factors for each vulnerability instance in the data set. Upon discussions with the security personnel and conducting sensitivity analyses using different combinations of the weights, we found the strategy of assigning equal weights to all the factors to be best for triaging the vulnerability instances. It is to be noted that an organization can fine-tune both the assignment of numerical scores and the preference for the factors in order to customize the VPSS to best suit its needs.

After the aforementioned measures, we now had a representative data set with all attribute information, in a natural language format, that provided all of the vulnerability instances along with their respective VPSS scores. To obtain important features and convert them into their respective numerical values, we applied natural language processing-based feature extraction techniques on this representative data set. In particular, we used the term frequency-inverse document frequency (TF-IDF) vectorizer to obtain

TABLE 5
A Sample List of Questions

Number	Question
Q1	Rating of the relevance of web and database servers.
Q2	Rating of the relevance of machines belonging to important stakeholders in the organization.
Q3	Rating of the relevant subnets/sub-domains in the network, based on asset criticality.
Q4	Number of analysts available and the personnel-hours allocated for cyber vulnerability management in a given time period.
Q5	Average numbers of vulnerabilities mitigated by the various types of analysts in a given time period.

TABLE 7
A Sample of Tunable Parameter Values for Random Forest Model

Tunable Parameter	Value Set 1	Value Set 2	Value Set 3	Value Set 4
n_estimators	200	200	200	200
max_depth	200	None	None	500
min_samples_split	4	2	2	2
ccp_alpha	0.02	0.01	0.02	0.00
MSE	**0.029**	**0.011**	**0.029**	**0.012**

TABLE 8
A Sample of Architectures Used for Deep Neural Networks

Hyperparameter	Value Set 1	Value Set 2	Value Set 3
Number of hidden layers	4	6	4
Number of neurons per layer	128, 256, 256, 256	512, 256, 128, 64, 32, 16	256, 128, 64, 64
Activation function	relu	relu	relu
Epochs	500	500	200
Batch Size	32	32	64
MSE	**0.0123**	**0.0120**	**0.0120**

important combinations of words. Using n-grams, we tried three values of n: 1, 2, and 3, which generated additional features in this training data set. A higher value of n may increase the accuracy of the model, but it also increases the computational costs. We did not find a significant difference in the MSE values when n was equal to 1 compared to higher values of n. With $n = 1$, an additional 11,589 features were generated for this training data set. Next, we developed supervised learning models by training on this data set with the VPSS score as the response (dependent) variable. In particular, we developed tree-based (random forest) and nonlinear (deep neural networks) supervised machine learning models due to the high sparsity found in this data set. We used Python programming language with Scikit-learn and Keras libraries to build and train these models. Table 7 (8) shows sample values of the tunable parameters (hyperparameters) considered for the random forest (deep neural networks) model.

4.2.2 Decision-Support System

Based on our conversations with the CSOC, we determined the values for the input parameters for the selection and the allocation models. The total number of personnel-hours (T in Algorithm 1) available on a weekly basis was set to 160 hours (equivalent of four full-time security personnel) for vulnerability mitigation. From the historical vulnerability mitigation data, the information about the types of vulnerability instances mitigated by each of the security analysts can be obtained. This information was learned from the historical vulnerability data and the appropriate adjustments/changes were made by the CSOC to establish the ground truth. Table 9 shows the four analyst skill types that were created based on the operating system (OS) expertise and familiarity with the network segment/host machines containing high-value assets. For instance, a senior analyst with expertise in using a Microsoft Windows OS and familiarity with the machine(s) containing the different types of asset (s) belonged to analyst skill type 1. Next, using text analytics, the various types of vulnerabilities in the data sets were categorized based on the OS and asset types, and used this information to match the analyst skill types to the vulnerability instances. Table 10 shows the different types of vulnerability instances based on the OS and the asset information. A perfect match between the analyst type and the vulnerability instance is shown under the 'Matching Vuln. Type' column in Table 9. Fig. 6 shows the total number of vulnerability instances per type found in the training data set. Correspondingly, Fig. 7 shows the estimated time required for mitigating these vulnerability instances per type by the matching analyst skill type (see Table 10). It is noted that there is a large number of vulnerability instances

TABLE 9
Matching Security Analyst SKill and Vulnerability Types

Analyst Skill Type	Expertise	Matching Vuln. Type
Type 1	Windows OS and critical-/high-value assets	1,3
Type 2	Linux OS and critical-/high-value assets	2,4
Type 3	Windows OS and medium-/low-value assets	5,7
Type 4	Linux OS and medium-/low-value assets	6,8

TABLE 10
Vulnerability Types

Vuln. Type	Asset and OS Type
Type 1	Critical-value asset and Windows OS
Type 2	Critical-value asset and Linux OS
Type 3	High-value asset and Windows OS
Type 4	High-value asset and Linux OS
Type 5	Medium-value asset and Windows OS
Type 6	Medium-value asset and Linux OS
Type 7	Low-value asset and Windows OS
Type 8	Low-value asset and Linux OS

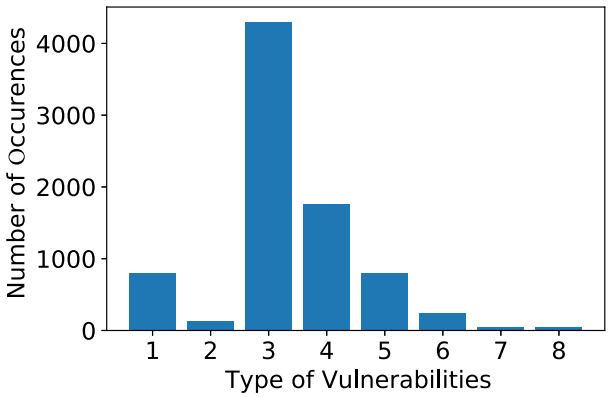

Fig. 6. Number of vulnerability instances per type.

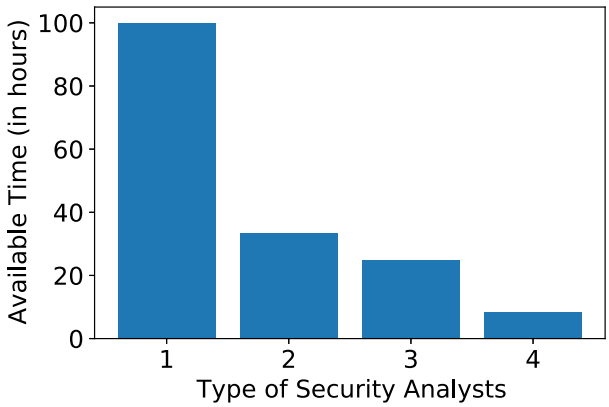

Fig. 8. Time available for vulnerability mitigation per analyst type.

found in machines running with a Microsoft Windows OS, which are in the same network segment that contains high-value assets (i.e., type 3). In our experiments, an additional 10% (x% in Algorithm 1) was added to the estimated mitigation time if the vulnerability instance type did not match the analyst skill type, in discussion with the CSOC. This number can be obtained from historical security personnel key performance indicator data or with the help of the CSOC manager (through a survey). This additional time value is used as a cost or a penalty that would force the optimization algorithm to minimize the sub-optimal pairings. This additional time comes into consideration when evaluating the various candidate solutions in Algorithm 1 to obtain a solution with minimum sub-optimal pairings. Fig. 8 shows the total (regular) time available for vulnerability mitigation per analyst type, per week (R_k in Algorithm 1). The sequential optimization algorithm was executed for 52 iterations on this data set (with 8031 vulnerability instances), where each iteration represented a one-week time period. The results were recorded for each iteration for all the methods (the proposed VPSS-based, CVSS value-based, KPI-driven, and VULCON), which were described in the previous section. Next, we explain the results obtained from the experiments.

5 Analysis of Results

In this section, we present the results obtained from the conducted experiments and provide an analysis of these results. First, we present the results for the ML-based VPSS scoring model, followed by the selection and allocation models of the decision-support system.

Tables 7 and 8 show the MSE values obtained using the tree-based and the nonlinear models, respectively. Based on the lowest MSE values obtained using the various configurations of each of these models, we selected the 'Value Set 2' configuration of the random forest approach to be the VPSS scoring model. It is to be noted that a non-zero cost complexity pruning parameter (ccp_alpha) value can help counter the issue of over-fitting (i.e., a low ccp_alpha value makes a model robust against some variations in the data samples, which should give a better performance by accurately estimating the VPSS scores of unseen data samples). Next, we present the results of the selection model and compare them with those obtained using the CVSS value-based, KPI-driven, and VULCON methods.

Fig. 9 shows a comparison of the performances of all the vulnerability selection methods over a 52-week period. The reported numbers are cumulative from the first iteration until the respective iteration number. It can be seen that the proposed approach using the VPSS-based selection model of the decision-support system is able to select vulnerability instances for mitigation with the largest cumulative vulnerability exposure score. This proposed model is shown to significantly outperform the KPI-driven method after the first 10 iterations, as the latter method is a greedy approach that focuses on selecting a greater number of vulnerability instances rather than selecting the ones that reduce the total vulnerability exposure of an organization. It can also be observed that the VPSS-based method outperforms the CVSS value-based and the VULCON methods throughout the 52-week period, which was the objective of the optimization model.

Fig. 10 shows the number of vulnerability instances selected from the web and database servers (pertaining to the relevance factor) using different methods. It can be seen that the VPSS-based selection model is able to outperform the other methods by selecting a greater number of vulnerabilities that are found in organization-specific relevant machines over the 52-week period. In particular, a greater number of vulnerabilities are prioritized from the web and database servers for mitigation using the proposed VPSS-based method when compared with the other methods. We

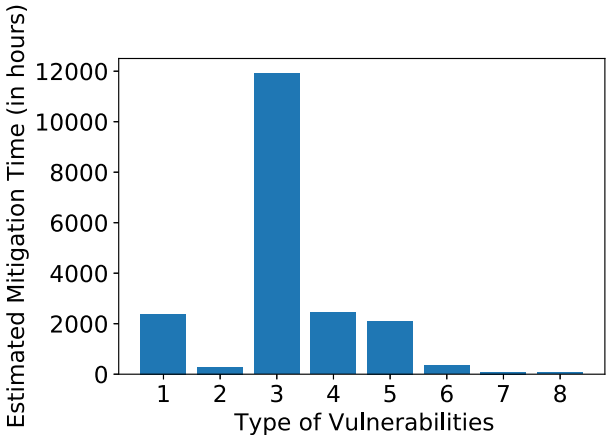

Fig. 7. Estimated mitigation time for vulnerability instances per type.

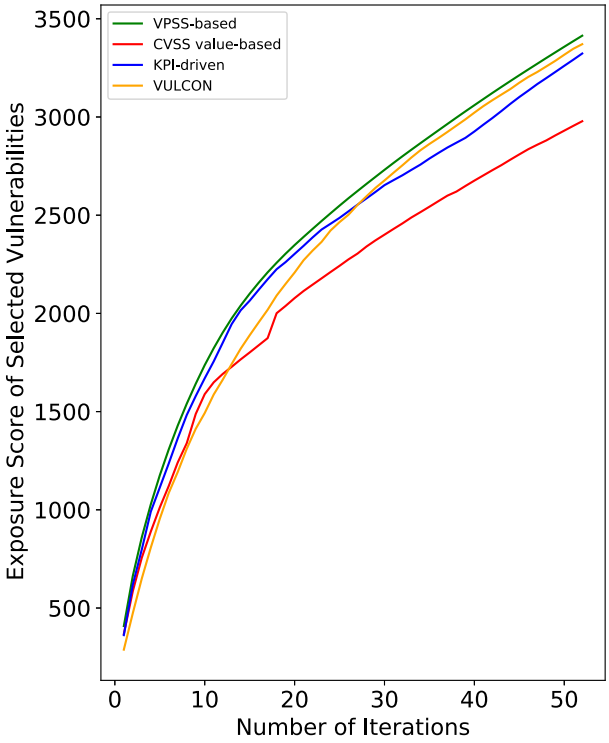

Fig. 9. Comparison of the exposure score of selected vulnerabilities.

note an interesting observation with the performances of the VPSS-based and the VULCON methods. In our vulnerability data set, some of the critical- and high-value assets were found on web and database servers. As a result, with VULCON's built-in feature to give a higher rank to vulnerabilities found on the critical servers (critical/high value assets), it selects a greater number of vulnerabilities that are found on these servers (as observed in Fig. 10, between the iteration range of 5 to 30). However, the objective of the proposed VPSS-based method is to select vulnerabilities for mitigation that reduce the organization's exposure score, which is maintained throughout the 52-week period (Fig. 9).

Fig. 11 shows the total number of vulnerabilities selected for mitigation for each method, which are found on machines with a low level of protection (see Table 6). In particular, the VPSS-based approach is able to prioritize

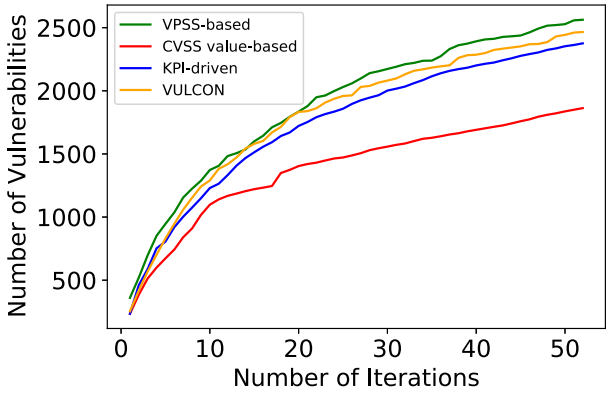

Fig. 11. Comparison of the number of vulnerabilities selected from machines with a low level of protection.

selection of vulnerabilities associated with machines running Windows OS 2008 (and older) and SQL version 2012 (and older). These machines, with limited or no support provided by the vendor, have less protection and are more susceptible to new cyber threats/attacks. It can be observed that the VPSS-based selection model performs the best among all of them, followed by the VULCON and the KPI-driven methods. This result points out one of the drawbacks of the CVSS value-based vulnerability selection method, in that vulnerabilities with low CVSS values that are found on more susceptible machines (with a low level of protection) in an organization's network are not selected for mitigation. Fig. 12 shows the total number of vulnerabilities selected from/in the vicinity of machines with critical- and high-value assets over the 52-week period. It was observed that the VULCON method, with the built-in feature of ranking vulnerabilities found on critical hosts higher than other vulnerabilities, performs as well as the proposed VPSS-based method. It can also be observed that the KPI-driven method selects a greater number of vulnerabilities than the VPSS-based method because many vulnerabilities pertaining to high-value assets had a lower estimated mitigation time associated with them (i.e., they belonged to bins 1 and 2in Table 3). It was also noted that the proposed VPSS-based method selected a greater number of vulnerabilities pertaining to the critical-value assets when compared to the KPI-driven method.

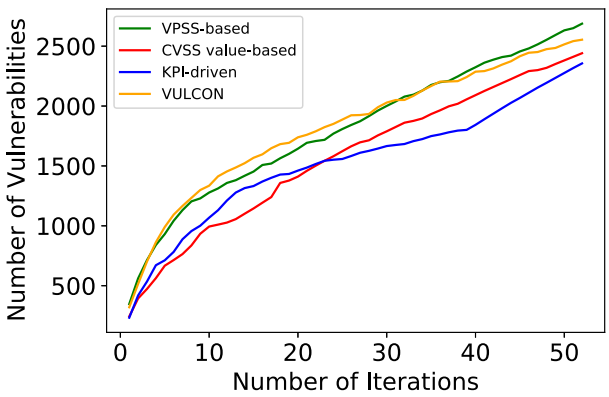

Fig. 10. Comparison of the number of vulnerabilities selected from organization-specific relevant machines.

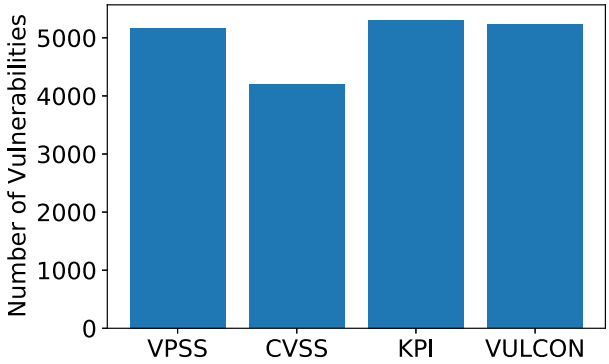

Fig. 12. Total number of vulnerabilities selected from high and critical value assets.

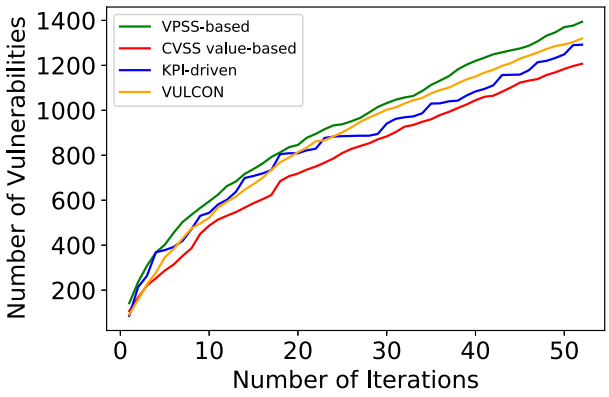

Fig. 13. Performance comparison on top host machines.

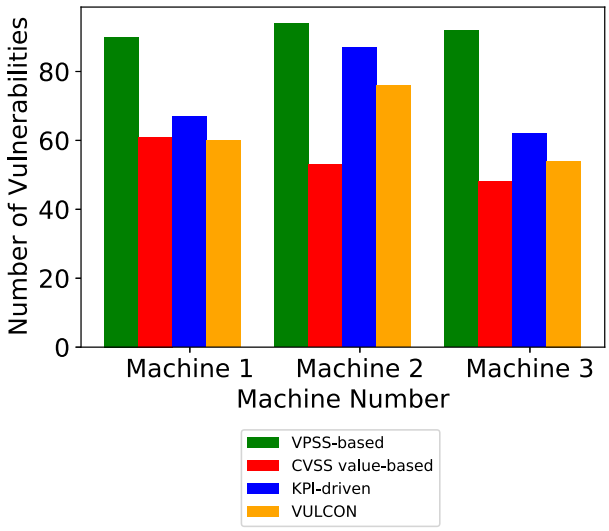

Fig. 14. Performance comparison on individual host machines.

Figs. 15a and 15b shows the comparisons in the performances of all the methods in terms of the number of vulnerabilities that were selected based on their CVSS severity ratings. The CVSS value-based method outperforms all the other methods by selecting a greater number of vulnerabilities with critical and high severity ratings. Another interesting observation was made when comparing the results obtained from the CVSS value-based and the VPSS-based methods in the first iteration. It was observed that there were four vulnerability instances whose CVSS values were nine. One of the vulnerability instances was found on a machine that was organizationally more relevant (database server) and required an estimated mitigation time of six hours. The other three vulnerability instances were found in other parts of the network on machines that were not recognized to be relevant and the cumulative estimated time required to mitigate all three of them was six hours. The output of the VPSS-based method contained the first vulnerability instance along with other context-sensitive vulnerability instances, while the output of the CVSS value-based method contained the other three vulnerability instances along with other vulnerability instances of high CVSS severity values (nine and over). This result points out another drawback of the CVSS value-based method of prioritizing vulnerabilities, in which organizational context is not taken into consideration. On the contrary, the VPSS-based method takes into account both the organizational context and the severity of a vulnerability, and selects the ones for mitigation that reduces an organization's vulnerability exposure score. We were able to identify similar interesting results and reported our observations to the security personnel, who were able to validate the inherent security risk associated with the selected vulnerabilities and the respective host machines.

To further understand the impact of our proposed approach on the security posture of the organization, we selected a subset of important host machines to compare the results of prioritization of vulnerabilities found on them using the various approaches. The selected host machines were identified as relevant machines (i.e., web servers, database servers, and important stakeholder machines) and they also belonged to sub-domains identified with high and medium level of asset criticality. Fig. 13 shows the number of vulnerability instances selected for mitigation from the top 40 host machines, which had the largest number of vulnerability instances reported over the 52-week period. The VPSS-based method is able to select a greater number of vulnerability instances and the model selects them sooner (on an average, at least two iterations faster) compared to

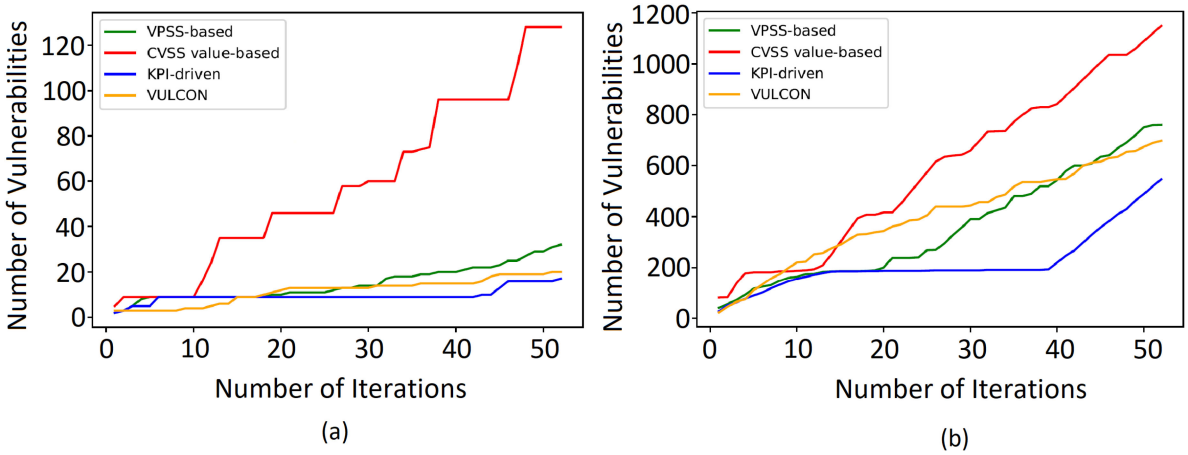

Fig. 15. Comparison of the number of vulnerabilities selected with (a) critical CVSS severity rating and (b) high CVSS severity rating.

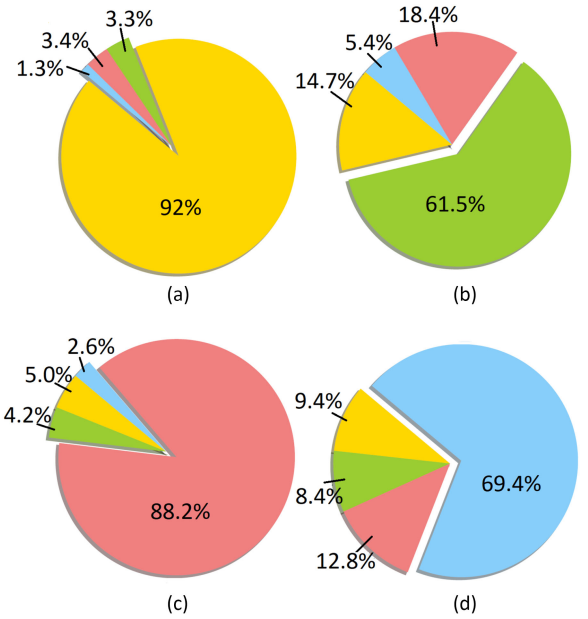

Fig. 16. Allocation of vulnerability types per analyst skill type: (a) skill type 1, (b) skill type 2, (c) skill type 3, and (d) skill type 4.

the other three methods. Fig. 14 shows the performance of all the methods on three individual sample machines (out of the 40 total machines). It can be clearly seen that the VPSS-based method outperforms the other methods in selecting a greater number of vulnerabilities for mitigation at an individual (important) machine level, which assists in strengthening the security posture of the organization.

The output of the allocation model of the decision-support system is the assignment of vulnerability instances to the available security analysts. The allocation model ensures that the number of matching pairs of vulnerability instance type and security analyst skill type (as per Table 9) is maximized. Figs. 16a and 16d shows the % of matching pairs per security analyst skill type for the entire duration of 52 weeks. It can be observed from Fig. 16a that 92% of vulnerability instances from type 1 and 3 were optimally allocated to security analysts from skill type 1. Fig. 16b shows that 61.5% of vulnerability instances allocated to analysts from skill type 2 were optimally matched as well (from vulnerability types 2 and 4). Similar observations were noted for analyst skill types 3 and 4, where a majority of the vulnerability instances allocated were optimally matched to the respective vulnerability types (per Table 9). It is to be noted that the decision-support system is iteratively executed, in which the selection model first selects the context-sensitive vulnerabilities and then the allocation model finds the matching pairs. Any sub-optimal allocation is due to the disparity in the time required to mitigate vulnerability instances of a particular type and the time available for mitigation by analysts with matching skill type in that iteration (week). As a result, we observe that in Fig. 16a, 1.3%, 3.4%, and 3.3% of vulnerability instances are sub-optimally allocated to analysts from skill type 1 instead of skill types 4, 3, and 2, respectively. Similarly, due to this disparity where less time was required to mitigate vulnerability instances of types 2 and 4, and more time was available for mitigation by analysts from skill type 2, the additional time was

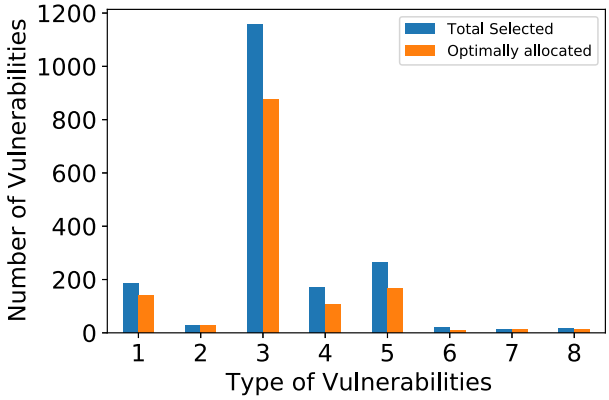

Fig. 17. Comparison of number of vulnerabilities selected and optimally allocated per vulnerability type.

allocated to mitigate vulnerability instances of other types. For instance, 14.7% of vulnerability instances allocated to analyst skill type 2 were from vulnerability types 1 and 3 due to a large number of them present in the data set). Fig. 17 shows the number of vulnerability instances that were selected for mitigation from the data set over the 52-week period and the number of vulnerability instances that were then optimally allocated among them, per vulnerability type. Experiment results obtained on other data sets were statistically similar.

6 Summary of Meta-Principles and Conclusions

The meta-principles obtained from this research study are as follows:

1) Given a natural language-based vulnerability data set, the combination of feature extraction, using the term frequency-inverse document frequency (TF-IDF) statistical measure and supervised learning with a Random Forest approach, is found to be an ideal choice for the development of a robust machine learning-based vulnerability priority scoring system (VPSS).
2) The VPSS-based vulnerability selection model for mitigation outperforms all the other methods that are currently employed at cybersecurity operations centers (CSOCs) and proposed in recent literature, in terms of selecting the context-sensitive vulnerabilities for mitigation to minimize the total vulnerability exposure score of an organization. The VPSS-based method takes into account both the organizational context and the severity of a vulnerability in order to prioritize vulnerabilities for mitigation. Because of the combined effect of the VPSS factors in the exposure score, an organization can be informed about the inherent security risk associated with the vulnerabilities selected for mitigation.
3) The drawback of the widely used common vulnerability scoring system (CVSS) is that the vulnerable host machines in an organization's network may remain unpatched for a long duration when the CVSS severity rating of the vulnerabilities found

in them is not high and they are, in turn, not prioritized.

4) The two-step sequential approach proposed for the decision-support system ensures that the context-sensitive vulnerabilities are selected for mitigation without putting any hard constraints on effectively matching the type of vulnerability instance and the skill type of the security personnel. The allocation model is executed after the selection model selects the optimal set of vulnerabilities for mitigation that reduces the vulnerability exposure score of an organization.

In this paper, we developed a novel advanced analytics and optimization framework for context-sensitive vulnerability management. In particular, we presented a machine learning-based vulnerability priority scoring system that prioritizes vulnerabilities for mitigation by taking both the organizational context and the severity of the vulnerabilities into consideration. We then presented a decision-support system, consisting of sequential optimization models for selection and allocation of vulnerabilities for mitigation. The experiments performed on the real-world vulnerability data sets demonstrated the superior performance of our approach in selecting context-sensitive vulnerabilities for mitigation, when compared with the currently employed vulnerability selection methods at the CSOCs and a recently proposed vulnerability prioritization method from literature. The security posture of an organization can be further improved by this proposed decision-support system, which optimizes the number of matching pairs of the types of vulnerability instances and the security personnel skill types for mitigation. Our framework is scalable and deployable at the CSOCs, and a paradigm shift in the way vulnerability management is performed today.

As a part of future work, correlation among vulnerabilities can be taken into consideration when selecting and allocating vulnerabilities for mitigation. A trade-off study comparing time-saving due to allocation of correlated vulnerabilities to the same security personnel and their potential burnout due to repetitive work could be conducted to further understand its impact. Another direction for future work is to develop methods for optimal staffing and scheduling of security personnel for vulnerability management in the wake of uncertain arrivals of vulnerability instances.

ACKNOWLEDGMENTS

The authors would like to thank the CSOC team for providing their vulnerability data to compare the various methods used in this research work.

REFERENCES

[1] "CVSS SeverityDistribution over time," 2020. Accessed: Feb. 11, 2021. [Online]. Available: https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time

[2] K. A. Farris, A. Shah, G. Cybenko, R. Ganesan, and S. Jajodia, "VULCON: A system for vulnerability prioritization, mitigation, and management," *ACM Trans. Privacy Secur.*, vol. 21, no. 4, pp. 1–28, 2018.

[3] Council of Economic Advisors, "The cost of malicious cyber activity to the U.S. economy," White House, Feb. 5, 2018. [Online]. Available: https://trumpwhitehouse.archives.gov/wp-content/uploads/2018/02/The-Cost-of-Malicious-Cyber-Activity-to-the-U.S.-Economy.pdf

[4] "Common vulnerability scoring system version 3.1: Specification document," 2020. Accessed: Feb. 11, 2021, [Online]. https://www.first.org/cvss/specification-document

[5] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0" in *Published by FIRST-Forum of Incident Response and Security Teams*, vol. 1. Cary, NC, USA: FIRST, 2007, Art. no. 23.

[6] C. Fruhwirth and T. Mannisto, "Improving CVSS-based vulnerability prioritization and response with context information," in *Proc. 3rd Int. Symp. Empir. Softw. Eng. Meas.*, 2009, pp. 535–544.

[7] L. Gallon, "On the impact of environmental metrics on CVSS scores," in *Proc. IEEE 2nd Int. Conf. Social Comput.*, 2010, pp. 987–992.

[8] G. Ridley, J. Young, and P. Carroll, "Cobit and its utilization: A framework from the literature," in *Proc. IEEE 37th Annu. Hawaii Int. Conf. Syst. Sci.*, 2004, p. 8, doi: 10.1109/HICSS.2004.1265566.

[9] H. Holm, T. Sommestad, J. Almroth, and M. Persson, "A quantitative evaluation of vulnerability scanning," *Inf. Manage. Comput. Secur.*, vol. 19, pp. 231–247, 2011.

[10] H. Holm, M. Ekstedt, and D. Andersson, "Empirical analysis of system-level vulnerability metrics through actual attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 6, pp. 825–837, Nov./Dec. 2012.

[11] H. Cavusoglu, S. Raghunathan, and W. T. Yue, "Decision-theoretic and game-theoretic approaches to it security investment," *J. Manage. Inf. Syst.*, vol. 25, no. 2, pp. 281–304, 2008.

[12] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Trans. Inf. System Secur.*, vol. 17, no. 1, pp. 1–20, 2014.

[13] A. Shah, K. A. Farris, R. Ganesan, and S. Jajodia, "Vulnerability selection for remediation: An empirical analysis," *J. Defense Model. Simul.*, vol. 19, no. 1, pp. 13–22, Jan. 2022.

[14] Z. Zhou and M. Cai, "Intersection signal control multi-objective optimization based on genetic algorithm," *J. Traffic Transp. Eng.*, vol. 1, no. 2, pp. 153–158, 2014.

[15] M. S. Ghanim and G. Abu-Lebdeh , "Real-time dynamic transit signal priority optimization for coordinated traffic networks using genetic algorithms and artificial neural networks," *J. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 327–338, 2015.

[16] N. Kwak and C. Lee, "A linear goal programming model for human resource allocation in a health-care organization," *J. Med. Syst.*, vol. 21, no. 3, pp. 129–140, 1997.

[17] M. Yoshimura, Y. Fujimi, K. Izui, and S. Nishiwaki, "Decision-making support system for human resource allocation in product development projects," *Int. J. Prod. Res.*, vol. 44, no. 5, pp. 831–848, 2006.

[18] S. Yao, Z. Dong, X. Wang, and L. Ren, "A multiobjective multifactorial optimization algorithm based on decomposition and dynamic resource allocation strategy," *Inf. Sci.*, vol. 511, pp. 18–35, 2020.

[19] L. C. e Silva and A. P. C. S. Costa, "Decision model for allocating human resources in information system projects," *Int. J. Project Manage.*, vol. 31, no. 1, pp. 100–108, 2013.

[20] Z. Lili, "An inverse optimization model for human resource allocation problem considering competency disadvantage structure," *Procedia Comput. Sci.*, vol. 112, pp. 1611–1622, 2017.

[21] F. A. Zaraket, M. Olleik, and A. A. Yassine, "Skill-based framework for optimal software project selection and resource allocation," *Eur. J. Oper. Res.*, vol. 234, no. 1, pp. 308–318, 2014.

[22] Y. Singh, A. Kaur, and B. Suri, "Test case prioritization using ant colony optimization," *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 4, pp. 1–7, 2010.

[23] R. T. Clemen and T. Reilly, *Making Hard Decisions With Decision-Tools*. Boston, MA, USA Cengage Learn., 2013.

[24] A. Shah, R. Ganesan, S. Jajodia, and H. Cam, "A two-step approach to optimal selection of alerts for investigation in a CSOC," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1857–1870, Jul. 2018.

[25] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*. Berlin, Germany: Springer, 1978, pp. 161–172.

[26] C. H. Papadimitriou, "On the complexity of integer programming," *J. ACM*, vol. 28, no. 4, pp. 765–768, 1981.

[27] T. Dumitraș and D. Shou, "Toward a standard benchmark for computer security research: The worldwide intelligence network environment (wine)," in *Proc. 1st Workshop Building Anal. Datasets Gathering Experience Returns Secur.*, 2011, pp. 89–96.

Soumyadeep Hore (Graduate Student Member, IEEE) received the BTech degree in mechanical engineering from the West Bengal University of Technology, Salt Lake, Kolkata, India, in 2015, and the MTech degree in industrial engineering from IIT (ISM) Dhanbad, India, in 2019. He is currently working toward the PhD degree with the Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, USA. His research interests include cybersecurity, predictive and prescriptive analytics, decision making under uncertainty, machine learning, multi-objective optimization, and deep reinforcement learning.

Fariha Moomtaheen received the bachelor's degree in computer science and engineering from the University of Dhaka, Bangladesh, in 2017, and the master's degree in computer science from the University of South Florida, Tampa, FL, USA, in 2021. Her research interests include cybersecurity, intrusion and forensics analysis, operating systems security, data science, and machine learning.

Dr. Ankit Shah (Member, IEEE) received the BS degree in computer science from Florida Atlantic University, Boca Raton, FL, USA, in 2001, the MS degree in operations research from George Mason University, Fairfax, VA, USA, in 2016, and the interdisciplinary PhD degree in information technology from George Mason University, Fairfax, VA, USA, in 2019. He is currently an assistant professor of industrial and management systems engineering with the University of South Florida (USF). He also holds a courtesy faculty appointment of an assistant professor with the Computer Science and Engineering Department. He is the director of Artificial Intelligence Research Laboratory for Secure Systems, USF and a member of the USF Institute for Artificial Intelligence + X. He is also a USF I-Corps fellow. Before joining USF, he was a researcher in the field of reinforcement learning with Lawrence Livermore National Laboratory, CA, and the Center for Secure Information Systems, VA. He has more than nine years of industry experience in the information technology and security sector. His research interests include the intersection of computer science, operations research, and information technology, with a strong focus on artificial intelligence for cybersecurity. His current research work is in the area of deep reinforcement learning and adversarial machine learning for threat reduction and secure systems.

Dr. Xinming Ou received the bachelor's and master's degrees from Tsinghua University in 1998 and 2000, respectively, and the PhD degree from Princeton University in 2005. He is currently a professor of computer science and engineering with the University of South Florida. Prior to that, he was a faculty member with Kansas State University from 2006 to 2015. His research is primarily in cyber defense technologies, with focuses on computer systems, programming languages, and human-centered approaches. He has broad interest and on-going work in cyber-physical system security, intrusion and forensics analysis, moving-target defense, and mobile system security. He is the author of the MulVAL attack graph tool which has been used by U.S. Idaho National Laboratory, Defence Research and Development Canada – Ottawa, North Atlantic Treaty Organization, U.S. National Institute of Standards and Technology, Thales Groups, General Dynamics, Johns Hopkins University Applied Physics Lab, Swedish Defence Research Agency, U.S. Army Research Laboratory, and researchers from numerous academic institutions world wide. His research has been funded by the U.S. National Science Foundation, Department of Defense, Department of Homeland Security, Department of Energy, National Institute of Standards and Technology, HP Labs, and Rockwell Collins. He was the recipient of the 2010 U.S. NSF Faculty Early Career Development (CAREER) Award, a three-time winner of HP Labs Innovation Research Program (IRP) Award, and 2013 Kansas State University Frankenhoff Outstanding Research Award.