VULCON: A System for Vulnerability Prioritization, Mitigation, and Management

KATHERYN A. FARRIS, Dartmouth College
ANKIT SHAH, George Mason University
GEORGE CYBENKO, Dartmouth College
RAJESH GANESAN and SUSHIL JAJODIA, George Mason University

Vulnerability remediation is a critical task in operational software and network security management. In this article, an effective vulnerability management strategy, called VULCON (VULnerability CONtrol), is developed and evaluated. The strategy is based on two fundamental performance metrics: (1) time-to-vulnerability remediation (TVR) and (2) total vulnerability exposure (TVE). VULCON takes as input real vulnerability scan reports, metadata about the discovered vulnerabilities, asset criticality, and personnel resources. VULCON uses a mixed-integer multiobjective optimization algorithm to prioritize vulnerabilities for patching, such that the above performance metrics are optimized subject to the given resource constraints. VULCON has been tested on multiple months of real scan data from a cyber-security operations center (CSOC). Results indicate an overall TVE reduction of 8.97% when VULCON optimizes a realistic security analyst workforce's effort. Additionally, VULCON demonstrates that it can determine monthly resources required to maintain a target TVE score. As such, VULCON provides valuable operational guidance for improving vulnerability response processes in CSOCs.

Categories and Subject Descriptors: C.2.0 Security and Protection **[D.4.6 Security and Protection]:** H.2.0 Security, Integrity, and Protection

General Terms: Cyber-Security Engineer, Vulnerability Data, Patch Management, Cyber-Security Operations Center (CSOC), Mixed Integer Constraint Optimization

Additional Key Words and Phrases: Cyber-security analysts, vulnerability triage and management, structured vulnerability response programs, cyber-security operations center (CSOC), multiobjective optimization

ACM Reference format:
Katheryn A. Farris, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. VULCON: A System for Vulnerability Prioritization, Mitigation, and Management. *ACM Trans. Priv. Secur.* 21, 4, Article 16 (June 2018), 28 pages.
https://doi.org/10.1145/3196884

Additionally, the authors are grateful to the Army Research Office (ARO) for supporting this work with MURI grant W911NF-13-1-0421, as well as the Air Force Research Laboratory and SMART scholarship for service for student support.
Authors' addresses: K. A. Farris, 920 Battelle Boulevard, Richland, WA 99352 (submitted while at Dartmouth College and finalized for publication while at Pacific Northwest National Laboratory [PNNL]); email: katheryn.farris@pnnl.gov; A. Shah, R. Ganesan, and S. Jajodia, Center for Secure Information Systems, George Mason University, Fairfax, VA 22030; emails: ashah20@masonlive.gmu.edu, {rganesan, jajodia}@gmu.edu; G. Cybenko, Thayer School of Engineering, 14 Engineering Dr., Hanover, NH, 03755; email: gvc@dartmouth.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 ACM 2471-2566/2018/06-ART16 $15.00
https://doi.org/10.1145/3196884

1 INTRODUCTION

Many organizations actively protect their enterprise-level networked assets from cyber attacks by periodically scanning for known vulnerabilities and implementing appropriate responses to the identified vulnerabilities. Vulnerabilities that do not have easy-to-implement or readily available patches need more creative solutions to manage or mitigate. Today, most cyber-security operations centers (CSOCs) are required to meet industry and federal standards for vulnerability management and remediation, depending on the type(s) of data stored (PCI 2018, NIS 2018). Because vulnerability scanning software outputs large, dense reports that are difficult for humans to parse and interpret as a whole, the total vulnerability exposure of an organization is not immediately evident. Consequently, the resources needed to manage vulnerabilities are not optimally allocated, resulting in high vulnerability exposures and suboptimal human resource allocation. Hence, a systematic and principled approach is needed to quantify total vulnerability exposure, prioritization, management, and mitigation.

In this article, a novel vulnerability management approach, named VULCON (VULnerability CONtrol), is introduced for addressing such challenges. To quantify key aspects of the vulnerability management problem, the following two performance metrics are defined for empirically measuring the efficiency of a CSOC vulnerability management process:

(1) Total vulnerability exposure (TVE): This metric "scores" the density of un-remediated vulnerabilities per month. A utility score is calculated for each unique vulnerability occurrence on a given host service. The scores are then aggregated across the current month's new vulnerabilities, combined with the residual vulnerabilities from previous months. This metric and its calculation are described in detail in Section 3.1.
(2) Time-to-vulnerability remediation (TVR): This metric defines the requirements for the maximum allowable number of months an organization is willing to tolerate a vulnerability persisting on the system. TVR is calculated as the difference between the time at which a vulnerability is detected and the time at which it is selected for mitigation. It is not realistically possible to address all vulnerabilities found by a scan of a large enterprise network between periodic scans. As such, TVR quantifies the organization's overall tolerance for the latencies between discovery and response.

The research objective of this article is to propose a comprehensive framework that (1) quantifies the factors that contribute to TVE; (2) formally describes and measures an organization's TVR to assess the latency between a vulnerability's discovery on the network to its time of remediation; (3) develops a multiobjective optimization model to determine the vulnerabilities to patch among the prioritized list that will optimize the above performance metrics, subject to resource constraints; and (4) uses the optimization to generate quantifiable reports and metrics based on which vulnerabilities can be remediated.

The primary contribution of the research is the development and evaluation of an efficient CSOC vulnerability management system, called VULCON. VULCON quantifies a vulnerability's severity, age, persistence, and asset's mission importance to the organization. VULCON calculates the TVE by using a utility function that combines the values of the above factors to determine a mitigation utility (MU) for each vulnerability. The MU score is then used to prioritize vulnerability instances (VIs) for mitigation (expanded explanation and mathematical definitions are provided in Section 3.1.1). The list of prioritized VIs is then optimized on and selected for mitigation based on a set of constraints (expanded explanation and mathematical definitions are provided in Section 3.1.2). This is used to prioritize vulnerabilities to mitigate under the constraints of limited human resources. Actually selecting vulnerabilities from the resulting prioritized list to optimally

reduce the TVE can be a hard combinatorial optimization problem. Additionally, VULCON defines and measures TVR by calculating the difference between the time at which the vulnerability is detected and the time at which it is remediated. By formally defining and measuring TVR, VULCON provides real-time feedback to the CSOC engineering and management teams on their remediation latencies, a critical feedback metric for understanding an organization's security performance.

The second key contribution of the article is the optimization algorithm that determines which vulnerability to select for mitigation from among the prioritized list of vulnerabilities. The above optimized selection is achieved by using a mixed-integer optimization formulation, which optimizes the performance metrics of TVE and TVR, subject to constraints on available personnel-hours. Other contributions include tradeoff analyses between the performance metrics and the meta-principles from the experimental results that provide the knowledge to set up and efficiently operate a CSOC's vulnerability management processes.

The article is organized as follows: Section 2 presents related literature; Section 3 describes the vulnerability management framework; Section 4 describes the vulnerability data and processing flow; Section 5 offers an overview of the experimental methodology and data used in this research; Section 6 provides the numerical experiments and their results, tradeoffs, scalability, and validation for the model, as well as a summary of meta-principles obtained from the above studies; Section 7 addresses challenges to internal and external validity; and Section 8 contains conclusions and directions for future research.

2 RELATED LITERATURE

This section describes related literature categorized into different domains. First, a review of recent research is provided in the area of vulnerability scoring, management, and treatment, as well as literature on data-driven solutions to vulnerability management. Then, recent work in triage metrics and optimization as it relates to other disciplines (i.e., emergency department [ED] medicine and condition-based maintenance) is summarized. Finally, literature and concepts from the foundational areas of multiobjective optimization are reviewed.

Vulnerability Scoring, Management, and Treatment. The Common Vulnerability Scoring System (CVSS) provides a base score for assigning vulnerability severity (Mell et al. 2007). It is defined by organizations such as Carnegie Mellon University's Computer Emergency Response Team (CERT) and the National Institute of Standards and Technology (NIST). CSOCs typically evaluate vulnerabilities by their CVSS base score. Allodi and Massacci (2014) find that incorporating external factors such as black market exploit data into the CVSS base score provides a more statistically significant indication of *true* vulnerability severity. Holm et al. (2012) indicate in their research that security modeling with CVSS data alone does not accurately portray the time-to-compromise of a system. They found that security models that base decisions on only the most severe CVSS data are less reliable than those that consider all vulnerabilities, regardless of their CVSS severity. To date, most CSOC vulnerability response programs that use CVSS typically amount to working down the list of vulnerabilities from the "most severe" to "least severe." As the aforementioned studies indicate, the CVSS base score is not necessarily enough to assess the *true* severity of a vulnerability, and, as such, VULCON incorporates more features in addition to the CVSS values (this is fully described in Section 3.1.1).

Two metrics related to this work have recently been proposed for assessing vulnerability exposure, namely, the median active vulnerabilities (MAVs) and vulnerability-free days (VFDs) (McQueen et al. 2009). These metrics were deduced from a vulnerability lifeline model based on measuring when a vulnerability is reported to a vendor to when that same vendor issues the patch. VULCON (and its corresponding experimental results) is different in that this body of research

investigates vulnerability management, prioritization, and the *true* state of vulnerability exposure based on live data from a CSOC. Basing an analysis on when the vendor discovers a new vulnerability to when it releases the patch is a good first step in modeling vulnerabilities. To complete this picture, more research is needed in the area of understanding how the vulnerabilities are actually prioritized, patched, and managed in an operational setting. Just because the vendor releases a vulnerability patch does not necessarily mean it is effectively deployed. Finally, as a movement toward research in vendor-to-CSOC relationships in vulnerability patch deployment, Cavusoglu et al. (2008) developed a game-theoretic model that incorporates a cost-benefit analysis of patch management to understand better methods of interaction between a vendor and a given CSOC.

Data-Driven Solutions in Vulnerability Management. Data-driven analysis in computer security has been fairly rare until recent years. NIST's most recent standards on data management were developed and written in their 2011 report on continuous monitoring in CSOCs (Dempsey et al. 2012). In the pursuit of science, some CSOCs are becoming more willing to share their data, along with proper data handling agreements, and large research organizations who own cyber-security data are becoming more interested in consolidating, anonymizing, and sharing their data for the mutual interest of moving the discipline of cyber-security forward. Some recent work in this area includes proposing a data-driven vulnerability maintenance policy, which uses Markov decision processes to generate and graphically evaluate relevant maintenance policies with limited data visibility (Afful-Dadzie and Allen 2014). Additionally, some work was achieved to develop research methods for autocorrelated vulnerability data. Afful-Dadzie and Allen (2016) use a hybrid moving centerline residual-based and adjusted demerit chart. The authors provide an analysis that directs an administrator to unusual cases when automated patching is insufficient.

Triage Metrics and Optimized Prioritization as It Relates in Alternative Disciplines. This analysis would not be complete without a rigorous review of other disciplines that developed metrics for prioritizing actions given resource constraints. In health care, EDs need to improve triage performance, and condition-based preventative maintenance leverages optimized scheduling.

In the discipline of emergency medicine, defined metrics are needed to improve ED operations and patient flow (Welch et al. 2011). Models such as "Lean thinking" use the methods known as value stream mapping, which diagrams and times the necessary process steps and physical layout improvements to make travel time and inventory tasks more efficient. Examples of how Lean thinking improved ED patient flow are that fewer patients sit in examination rooms while lab or test results are run; patients now wait in a "disposition area" to be discharged (Holden 2011). Another well-known example of optimizing ED patient flow is on minimizing wait times for patient care (Horwitz et al. 2010). Horwitz et al. captured ED performance based on the key metrics to devise a standardized and recommended timestamp patient processing flow to reduce variation across hospitals.

Finally, the reliability engineering, systems engineering, and systems safety communities contribute to the development of condition-based maintenance, a methodology for prioritizing and executing proper triage of mechanical failure. Historically, two methodologies for maintenance exist: (1) time-based maintenance (e.g., routine vehicle adjustments, oil changes, and tuning) and (2) condition-based maintenance (e.g., vehicle adjustments, oil changes, and tuning based on data that indicates *when* the vehicle needs it, effectively achieving more targeted and precise maintenance scheduling) (Asadzadeh and Azadeh 2014).

Concepts such as the "Lean thinking" approach to making the inventory task more efficient in ED patient flow and condition-based maintenance provide tremendous inspiration for new developments in the area of CSOC vulnerability maintenance and management policies.

Optimization Algorithms and Models. Many real-world problems have multiple and often conflicting objectives. Such problems have tremendous practical importance as they produce a set of solutions based on the importance assigned to each of the objectives. Critical insights can be obtained about the relationships of the parameters (of the objectives) by studying the tradeoff (Pareto-optimal) solutions. One of the ways to solve such problems is by using the goal programming method (Rardin 1998; Ignizio 1983). In this method, the weights are assigned to the goals (objectives) indicating their relative importance and the solution is obtained by minimizing deviations from the target values of the goals using methods such as weighted sum, lexicographic, and weighted min-max.

The goal programming method is explained further in Section 3.1.2. Valuable insights are offered and multiple objectives achieved through various studies in published literature. Selecting supplies while minimizing the cost of purchasing a number of rejected items in manufacturing industries (Jadidi et al. 2014), selecting stocks/assets that maximize the expected return and minimize loss in financial portfolio management (Aouni et al. 2014), and tradeoff studies between staffing flexibility, costs, and selection of equipment investments in a bank are examples from the literature where real-world problems with conflicting objectives have been studied.

Cyber-Insurance. As the demand for cyber-insurance programs increases, so too does the need for dependable metrics on the overall security posture of the organization's enterprise network (Biener et al. 2015). One of the challenges facing cyber-insurance programs is defining industry standards for what constitutes an acceptable level of organizational exposure to cyber-threat actors. To advance network security in a manner by which these standards can be quantified, we have to scientifically define which vulnerability features bear the greatest influence on overall "threat." VULCON is a scientific process aimed at such challenges. Finally, as CSOCs are required to comply with data protection standards and laws (e.g., Health Insurance Portability and Accountability Act (HIPAA); Payment Card Industry, Data Security Standard (PCI DSS); and the Federal Information Security Management Act (FISMA)), the development and maturation of vulnerability response programs further supports the relevancy of these efforts.

3 THE VULCON VULNERABILITY MANAGEMENT FRAMEWORK

VULCON is a framework for CSOC vulnerability prioritization and management. VULCON is intended to operate on a periodic basis by ingesting vulnerability scan reports and analyzing them in the context of previous scans, the organization's mission, and resources. For each iteration, multiple raw scan data files are merged and preprocessed into a clean, unified representation of the empirically observed vulnerability status of the organization's networked assets.

The VULCON framework is depicted in Figure 1. The blue boxes represent consumers and users of VULCON, including managers, operators, engineers, and analysts. The green boxes represent data preprocessing and the VULCON optimization algorithm that recommends a specific prioritization of vulnerabilities based on their calculated MU scores (a full description is provided in Section 3.1.1). The yellow boxes represent the devices on which the vulnerability scans are run and controls that either can be or already are implemented. The orange boxes are VULCON inputs that include data about the organization's mission-critical services, NIST's National Vulnerability Database (NVD), the periodic Nessus scan reports (in this work, the periodicity is monthly, which is used hereafter), and organizational performance requirements and constraints. The gray boxes represent outputs from VULCON's preprocessing and optimization code. These outputs provide security exposure metrics and vulnerability management plans to managers, operators, analysts, and engineers.

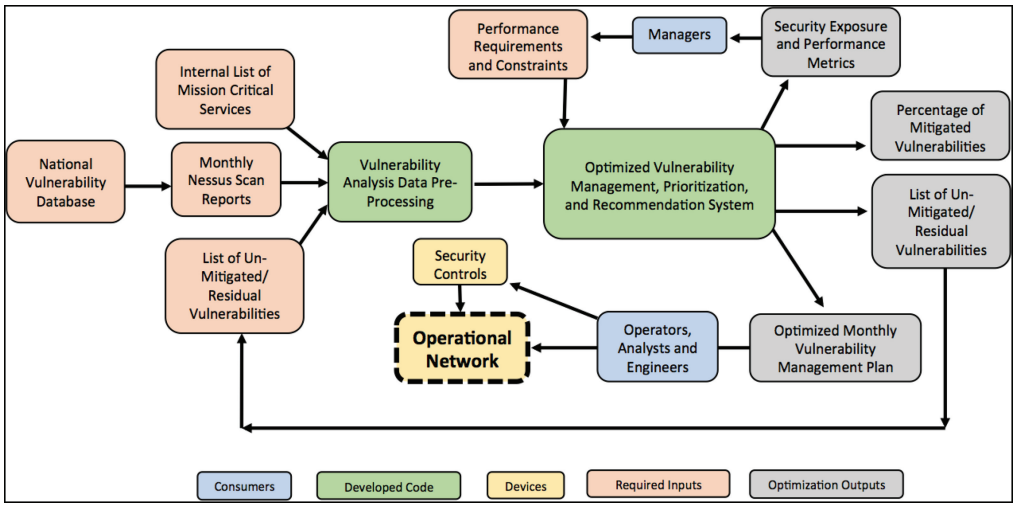

Fig. 1. The VULCON framework for vulnerability processing and management flow.

In Section 3.1, the concept of a TVE score is introduced. It is a fundamental metric that quantifies the utility scores for the total set of vulnerability occurrences on a given unique host service per month, thereby providing a metric for a CSOC to benchmark the vulnerability exposure.

3.1 Computation of Total Vulnerability Exposure

The TVE metric is a fundamental output of VULCON. It is a quantitative value that informs CSOCs of their monthly vulnerability exposure, offering a comprehensive, data-driven, actionable metric to inform decisions about personnel allocation and report metrics on their current security posture to management.

The TVE score is computed in the following three steps:

(1) Compute the MU of each VI (see Section 3.1.1).
(2) Solve a mixed-integer linear program to select VIs for mitigation in a time period, with the objective of minimizing the aggregate unmitigated MU (see Section 3.1.2).
(3) Compute the TVE as the sum of the unmitigated MU values (see Section 3.1.3).

The remainder of this section describes the qualitative attributes required for the above three steps that VULCON performs.

As shown in Figure 1, VULCON takes the following inputs:

—Periodic (in VULCON's experiments, monthly) vulnerability scan data (such as that provided by Nessus, for example, discussed in more detail in Section 4.1); mission-critical services as determined by the system administrators and the organization's mission; data from the NIST NVD; and internal scan reports
—The previous time period's unremediated (residual) VIs, where applicable
—The target organizational TVR requirements
—The target personnel-hour constraints
—The target organizational TVE score requirements

3.1.1 Step 1: Computing MU. VULCON calculates a VI's MU (u_i) using the following input values:

— i is the VI index, measured as a unique tuple $< v, s >$, such that v represents the unique NVD vulnerability and s represents the specific system on which that vulnerability was found in the scan. Note that s is uniquely determined by the given host/IP address, port, and protocol and $1 \leq i \leq I$, where I is the total count of VIs in the scan.
— s_i: This is the associated severity score of the underlying vulnerability arising in the VI, where $1 \leq s_i \leq 10$. This value is obtained from the CVSS,[1] where 1 represents the least severe score and 10 represents the most severe. This variable is normalized to a scale between 0 to 1 before it is input into the weighted sum formula.
— p_i: This is the number of months (or scan periods) a VI has persisted, namely, the number of consecutive months a VI has been present in scans. This variable is normalized to a scale between 0 and 1 before it is input into the weighted sum formula.
— a_i: This is the chronological age of the underlying vulnerability arising in a VI. Vulnerability age is computed directly from the Common Vulnerability Enumeration (CVE) code. For instance, the code "CVE-2013-1119" means that the vulnerability has been publicly known since the year 2013. It is derived that the vulnerability is 3 years old as of the year 2016. This variable is normalized to a scale between 0 and 1 before it is input into the weighted sum formula.
— Q_i: These are scalar values specified by the system operators to indicate the mission criticality of the corresponding host service arising in a VI.
— α, β, and γ: These are system-operator-defined weights used to tune the definition to meet enterprise needs.

The weighted (by α, β, and γ) sum of the above attributes for a given VI is the raw utility value defined as

$$v_i = \alpha * s_i + \beta * p_i + \gamma * a_i. \tag{1}$$

The values s_i, p_i, and a_i are transformed into equivalent unit norms of 0 to 1 so that they can be combined in the weighted sum formula. Next, VULCON accounts for mission impact and criticality. As shown in Figure 1 and defined above, VULCON collects a mission criticality score, Q_i, for the host service, arising in the VI. VULCON multiplies each raw utility score by the scalar value Q_i to get the MU, u_i, value for each VI:

$$u_i = Q_i * v_i. \tag{2}$$

3.1.2 Step 2: VULCON Optimization. Step 1 computes the MU value (u_i) of each VI. By factoring in a VI's severity (s_i), persistence (p_i), age (a_i), and mission criticality (Q_i), MU captures the urgency with which a CSOC should address each individual VI.

Sorting VIs by their MU values produces a prioritized list of vulnerabilities for CSOC staff to manage and mitigate, but prioritization alone does not fully capture the tradeoffs between reducing the CSOC's vulnerability exposure against available resources. For example, mitigating a small number of VIs with a high MU may not be as effective as mitigating a large number of VIs with lower-valued MU—the high-valued VIs might require a huge human effort to mitigate, which could be applied to many lower-valued VIs. The challenge is to reduce the overall vulnerability exposure (i.e., TVE) as much as possible, while respecting CSOC's resource constraints.

[1]The CVSS score is the de facto industry standard, and it is readily available from the NIST website. Indeed, other severity scoring systems exist such as Microsoft's Exploitability Index (Mic 2017) and Symantec's Threat Severity Assessment (Sym 2017). While the experiments described herein use CVSS values, the end-user can utilize any of the alternative severity scoring metrics in Equation (1).

To model this tradeoff, VULCON uses a mixed-integer programming formulation for the optimization problem. It is solved using the goal programming method, a branch of multiobjective optimization (Rardin 1998; Ignizio 1983). In goal programming, one or more solutions are found that either satisfy a set of goals or minimally violate them. The decision maker selects a target value as input for each goal. A deviation variable is assigned to each goal, thereby representing the magnitude by which the goal target is not achieved. Higher-penalty coefficients (weights) are assigned to the deviation variables of goals that receive a higher priority, according to the decision maker's needs. By choosing the value of the above weights, an artificial penalty function is constructed. The objective of goal programming is to find a solution in the decision space that minimizes the deviations from the criterion space of the goals that are modeled as constraints.

The parameters of VULCON's optimization problem are:

- i is the VI index, measured as a unique tuple $< v, s >$, such that v represents the unique NVD vulnerability and s represents the specific system on which that vulnerability was found in the scan. Note that s is uniquely determined by the given host/IP address, port, and protocol and $1 \leq i \leq I$, where I is the total count of VIs in the scan.
- c is the constraint index, with $c \in C$.
- H is the total number of personnel-hours available between scans.
- R_i is an upper bound on the number of months VI can persist before being removed. This is specified by organization policy as the *required* TVR.
- h_i is the estimated number of personnel-hours required to mitigate VI.
- u_i is the MU score for VI.
- p_i is the total number of months VI has persisted. This is referred to as the *current* TVR.
- S is the target TVE score, which can assume any nonnegative value and is met when there are zero deviations in a candidate solution.
- w_c is the assigned weight for deviations on constraint c.
- m is the maximum number of months any vulnerability has persisted in the dataset.

The VULCON model variables are:

- x_i, which is a binary 0-1 indicator variable with value 1 if the VI is selected for remediation and 0 otherwise.
- d_c is an integer variable that represents the amount of deviation in constraint c.
- e_i is the deviation variable used in the TVR constraint. It takes a value of 1 if the VI violates the TVR constraint, and 0 otherwise.

The goal programming objective function is the total penalty for deviating from the stated goals and constraints. We begin by listing the various goals and constraints and then describe the objective function thereafter.

The constraint for target personnel-hours is

$$\sum_{i=1}^{I} h_i * x_i \leq H + d_1. \tag{3}$$

The constraints for the target TVR value are given by the following:

$$1 + (p_i - R_i) \leq m * (x_i + e_i) \ \forall \ i, \tag{4}$$

$$x_i + e_i \leq 1 \ \forall \ i, \tag{5}$$

$$\sum_{i=1}^{I} e_i = d_2, \tag{6}$$

where m is set as the maximum number of months any vulnerability has persisted in the dataset. It should be noted that d_2 is a count of the number of times that a VI violated the TVR constraint.

The concept behind Equation (4) is that it forces the vulnerability to be selected, in which case $x_i = 1$, or for the violation variable of that vulnerability to be selected, in which case $e_i = 1$, when vulnerability persistence (p_i) equals the upper bound allowed on the persistence value (R_i).

Additionally, Equation (5) allows at most one of them to be selected for each vulnerability, and Equation (6) counts the number of total violations across all the remaining unmitigated vulnerabilities.

Finally, the constraint for the target TVE score is given by the following inequality:

$$\sum_{i=1}^{I}(1 - x_i) * u_i \leq S + d_3. \tag{7}$$

This effectively sums the total residual (unmitigated) mitigation utility values and compares it with the target value, S, adding d_3 if the inequality is violated.

The final VULCON objective function is

$$Z = \min \sum_{i=1}^{3} w_c * d_c. \tag{8}$$

Under the trivial situation with no deviations (i.e., $d_1 = d_2 = d_3 = 0$), the objective is to maximize the mitigated utilities as follows:

$$Z = \max_{x_i \in \{0,1\}} \sum_{i=1}^{I} u_i * x_i, \tag{9}$$

subject to $d_i = 0$, which is known to be satisfiable in that case.

The constraint $d_1 = 0$ imposes that the total available personnel-hours (H) must be greater than or equal to the sum of personnel-hours per vulnerability observation (h_i), which is mathematically formulated as

$$\sum_{i=1}^{I} h_i * x_i \leq H. \tag{10}$$

The second constraint, $d_2 = 0$, requires that the current TVR (p_i) must be less than the organization's required TVR (R_i), mathematically represented as[2]

$$p_i \leq R_i. \tag{11}$$

Finally, the third constraint, $d_3 = 0$, requires that the organization's current TVE score, defined as the sum of unmitigated MU values ($\sum(1 - x_i) * u_i$), must be less than or equal to the organization's target TVE score (S), based on the following mathematical formulation:

$$\sum_{i=1}^{I}(1 - x_i) * u_i \leq S. \tag{12}$$

[2]It should be noted that Equation (11) does not derive directly from Equation (4), if one sets $d_2 = 0$. This is because there is an underlying assumption that if e_i and m are removed, then the corresponding equation obtained will be $1 + (p_i - R_i) <= x_i$, which means $p_i <= R_i$ as x_i can take a value of either 0 or 1.

3.1.3 TVE Calculation. VULCON's TVE score calculation is one of the most pertinent outcomes for a CSOC, and it is measured as the sum of the most current set of unmitigated MU values in a CSOC's vulnerability report. For the purposes of this study, the TVE score is recorded at the beginning of every month, and is achieved by combining the following:

(1) The set of unmitigated MU values from the previous month's optimization output. This set of unmitigated MU values effectively represent the residual vulnerabilities that were not selected for remediation.
(2) The new month's set of unmitigated MU values from all vulnerabilities.

Before the data in the aforementioned steps (1) and (2) are combined, the persistence values of the data in step (1) are increased by a value of 1 and the MU values are updated accordingly.

VULCON's TVE score is a fundamental metric by which a CSOC can compare overall impact and improve the organization's security posture. Additionally, it provides a useful metric for reporting improvements up the management chain. The next section provides the algorithm behind VULCON.

3.2 VULCON Optimization Algorithm

VULCON's optimization steps are described in Algorithm 1, making note of its inputs, outputs, and requisite steps to get from the preprocessed monthly input to the final monthly output. Additionally, Sections 3.2.1 and 3.2.2 describe the monthly implementation and the computational complexity.

Several experiments were conducted to allow deviations only in some of VULCON's goals (constraints) at a time. A very large weight was assigned to the deviation variable of the desired goal

ALGORITHM 1: VULCON Optimization Algorithm

Input: Number of available personnel-hours, H, upper bound on TVR, R_i, total number of vulnerabilities, I, estimated personnel-hours, h_i, utility score, u_i, and persistence value in months, p_i, for each vulnerability, target TVE score, S, and weights (penalty coefficients), w_c, on the deviations from the goals.

Output: List of vulnerabilities, x_i, that are selected for remediation, and updated performance metrics, and personnel-hours with deviations

Step 1: Initiate the solution search for the objective of minimizing the penalties on multiple goals in the solver ($\sum_{c=1}^{3} w_c * d_c$)

Step 2: Solve the mixed integer programming model (solution search) to obtain the list of vulnerabilities x_i using the solver

Step 2a: For each list of vulnerabilities obtained for remediation during the solution search, verify the following constraints within the solver

for *each vulnerability (x_i),* **do**
    check the target TVR constraints (Equations (4), (5), (6)) ;
end

check the target personnel-hours constraint (Equation (3));

check the target TVE score constraint (Equation (7));

Step 2b: Perform optimality check within solver

if *an optimal solution is found;*

then
    stop the solution search in the solver;
end

return *List of vulnerabilities for mitigation, vector X with all x_i, number of deviations on all the goals*

to be held constant, whereas the goals that could be violated were assigned smaller weights for their deviation variables.

Finally, VULCON produces outputs that provide a CSOC with data-driven, actionable metrics, allowing them to quantify and communicate their current exposure both within the organization and up the management chain. These outputs include

—the list of vulnerabilities that were remediated,
—the list of vulnerabilities not remediated (i.e., "residual" vulnerabilities),
—updated monthly performance metrics on the organization's TVE and TVR requirements, and
—updated personnel-hours with their deviations, where applicable.

The next section discusses how the optimization algorithm is implemented on the month-to-month basis.

3.2.1 Month-to-Month Implementation. VULCON quantifies a vulnerability's severity, age, and persistence and the asset's mission importance to the organization on a month-to-month basis. Each month, VULCON calculates a new utility score for each VI based on the incoming monthly data. VULCON reads in the first month's vulnerability scan data and merges it with the CSOC's internal list of mission-critical services. It then quantifies and normalizes variables such as the vulnerability's severity (s_i), persistence (p_i), and age (a_i). VULCON's algorithm then iterates on each month's new set of data. For each month the algorithm iterates, the persistence score is increased by a value of 1 for all residual (unmitigated) vulnerabilities from the previous month. The TVE score (as described in Sections 3.1.1, 3.1.2, and 3.1.3) is reported at the beginning of each month by summing across the MU values of the combined set of residual vulnerabilities, plus the new month's vulnerabilities.

A more detailed explanation of how mission-critical services are defined and the rich features contained in the monthly vulnerability scan reports is provided in Section 4.1.2.

3.2.2 Computational Complexity. In reviewing computational complexity, two primary prongs for this analysis surfaced: (1) implementation of the data-merging and preprocessing steps, the MU calculation, and prioritization and (2) implementation of the multiobjective optimization model on the data. For the purposes of this section, the software used and machine specifications are described. See Section 6 for a complete description of experimental outcomes.

The data-merging, preprocessing, and utility score calculations were achieved using the programming software R, specifically the *base* and *dplyr* programming packages. All iterations took under 3 minutes to perform in total with the R software version 3.3.1 on a 3GHz Intel Core i7 machine with 16 GB RAM.

The multiobjective optimization model in this research used a mixed-integer programming method. Integer programming, in general, is in the NP complexity class (Papadimitriou 1981). The closest known problem that can be reduced to the algorithm used in this research is the knapsack problem. Selection from a prioritized list of vulnerabilities to minimize the TVE score while not exceeding the personnel-hours is a nontrivial problem. A decision problem, similar to the decision problem in knapsack, could be posed as: Can a particular TVE score be attained without exceeding a given personnel-hour value? The knapsack decision problem is nondeterministic polynomial-time complete, and the problem presented in this article is as hard as the knapsack problem. No known polynomial-time algorithm exists to solve the knapsack problem to obtain insights on the vulnerability management research problem. Hence, to study the interdependencies among the performance requirements and resource constraints, a goal programming mathematical model was formulated. Experiments were conducted where deviations from goals were only

allowed on some goals at a time to study the impact on the solutions obtained. The computational times to obtain solutions from the optimization model were all under 10 minutes on a 192GB RAM, 2.60GHZ dual-core machine using Julia and the Gurobi solver.

4 VULNERABILITY DATA AND PROCESSING FLOW

A CSOC's vulnerability management processing flow and structure can significantly influence the outcome of how efficiently personnel-hours and resources are allocated. Investigating a few vulnerabilities on a few hosts seems simple, but such methods do not scale up. When an administrator has to assess which vulnerabilities to remediate, mitigate, and manage first among hundreds of vulnerabilities, spanning across multiple thousands of hosts, the complexity increases exponentially. The CSOC has to prioritize, triage, and allocate resources in a manner that considers factors such as which host service the vulnerability appears on, the age of the vulnerability, how long it has persisted, and the vulnerabilities' severity.

This section introduces the data used in VULCON. What makes this work unique from other cyber-security research is that the datasets are real (as opposed to simulated or synthetically produced) and fully capture the network behavior that came from an operational CSOC. This body of research and VULCON experiments benefited from researchers being embedded in a CSOC's day-to-day operations for over a year. The valuable discussions that stemmed from this experience informed VULCON's design and helped determine the values of the parameters for the experiments. Additionally, public requests for more data from operational CSOCs date as far back as 2009 (Camp et al. 2009). While some progress has been made with datasets such as the Worldwide Intelligence Network Environment, the computer security research is disproportionately based on simulations and synthetically produced data (Dumitras and Shou 2011). Having had the ability to test VULCON on real data from a CSOC enriches the range of results and scientific insights that can now be provided to the cyber-security community.

Sections 4.1, 4.1.2, and 4.1.3 provide an overview of the data, its source, and preprocessing methods.

4.1 Data Description

Data from a Nessus vulnerability scanner was ingested and experimented upon for the VULCON methodology. Section 4.1.2 describes the specific Nessus software information. Ten sets of scan data from September 2015 to June 2016 were used for the VULCON experiments. The data spanned a little over 2,000 hosts and has 11 different attributes, which characterize each unique vulnerability occurrence on a unique host. Figure 2 displays a sample of what the raw vulnerability data looks like (nes 2017).

The Nessus vulnerability output has the following 11 attributes:[3]

(1) **Plugin ID:** A software code, each with a unique ID number, that identifies what attributes and vulnerabilities it is scanning for.
(2) **CVE code:** A system for labeling the most common vulnerabilities with a uniquely identifiable number.
(3) **CVSS:** The CVSS value provides a base score for assigning vulnerability severity.
(4) **Risk Level:** A one-word description to annotate vulnerability severity. The software vendor typically uses descriptors such as "Critical," "High," "Medium," "Low," or "Info." The CVSS value informs the risk-level description for most software vendors.[4]

[3]The VULCON experiments utilized the first seven features.
[4]The terminology "risk level" comes directly from the Nessus software output and is not a term defined by this research. The security software industry generally correlated CVSS values to an associated "risk level."

Plugin ID	CVE	CVSS	Host	Protocol	Port	Name	Synopsis	Description	Solution	Plugin Output
11849	CVE-2003-0831	9	192.168.150.131	tcp	21	ProFTPD file Transfer Newline Character Overview	Arbitrary code may be run on the remote server.	The remote host is running a version of ProFTPD which seems to be vulnerable to a buffer overflow when a user downloads a malformed ASCII file. An attacker with upload privileges on this host may abuse this flaw to gain a root shell on this host. ***The author of ProFTPD did not increase the version number ***of his product when fixing this issue, so it might be false.	Upgrade to ProFTPD 1.2.9 when available or to 1.2.8p.	
35777	CVE-1999-0502	10	192.168.150.131	tcp	22	Default Password (toor) for 'root' Account	An account on the remote host uses a known password	The account 'root' on the remote host has the password 'toor'. An attacker may leverage this issue to gain total control of the affected system.	Change the password for this account or disable it.	It was possible to execute the command 'id' on the remote host: uid=0(root)gid=0(root)groups=0(root)
10061	CVE-1999-0103		192.168.150.132	udp	7	Echo Service Detection	An echo service is running on the remote host.	The remote host is running the 'echo' service. This service echoes any data which is sent to it. This service is unused these days, so it is strongly advised that you disable it, as it may be used by attackers to set up denial of services attacks against this host.	-Under Unix systems, comment out the 'echo' line in /etc/inetd.conf and restart the inetd process -Under Windows systems, set the following registry key to 0: HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpEcho HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpEcho Then launch cmd.exe and type: net stop simptcp net start simptcp To restart the service.	

Fig. 2. A sample of what a typical Nessus vulnerability output file looks like. Example information was sourced from https://www.tenable.com/blog/new-nessus-feature-added-csv-export.

(5) **Host Name/IP Address:** The host name is a label that identifies the machine or device connected to the network (e.g., www.google.com). The IP address is a string of numbers that uniquely identifies a machine or device that is connected to the network (e.g., 172.16.254.1). Any machine or device may have both an IP address and host name, or just an IP address.
(6) **Protocol:** The method of communication and set of rules used by the designated host, machine, or device to "talk" to another host, machine, or device. TCP and UDP are example protocols.
(7) **Port Number:** An identifier for the endpoint of communication on the host, machine, or device that is determined by a logical construct.
(8) **Vulnerability Name:** A brief title of the vulnerability. It is fewer than five words and chosen by the software vendor.
(9) **Vulnerability Synopsis:** A short, one-sentence explanation of the vulnerability.
(10) **Vulnerability Description:** An expanded and more comprehensive explanation of the vulnerability. May provide current and background information. Can be anywhere from 20 to 100 words in length.
(11) **Solution:** Recommendations to patch the vulnerability. Examples include "Disable SSLv3" and "Contact the software vendor for this application."

4.1.1 Personnel-Hour Estimations. The estimated number of personnel-hours required to remediate vulnerabilities is not captured in the Nessus vulnerability scan output. Subject matter expert interviews with the security engineers were performed to obtain this data, whereby vulnerabilities were binned into one of three possible categories for the estimated number of required hours to remediate, namely:

—1 to 3 hours
—3 to 6 hours
—6 to 9 hours

The security engineers searched for keywords in the description such as "weak cipher," "default password," "configuration changes," or "password update" and binned those vulnerabilities as requiring 1 to 3 hours to remediate. The security engineers also estimated that any vulnerability description that required a *version upgrade* was automatically binned into the personnel-hour category of 3 to 6 hours, and 6 to 9 hours if the recommended remediation required a *system upgrade*. Finally, to define a unique personnel-hour value to each VI, the value was randomly selected between the upper bound and the lower bound for each estimated range. An example of this data output is represented in Table 1.

4.1.2 Data Source and Software. The Nessus vulnerability scanner version 5.0.1 generated the data. Nessus is developed and maintained by Tenable Security Solutions (nes 2017). The vulnerability scans are run on an organization's data center. Known as the "brains" of an organization, a data center centralizes the storage, management, and dissemination of data that is fundamental to an organization's computer and network operations. It can exist as either a virtual or a physical service.

4.1.3 Data Preprocessing. Before any optimization can be performed on the data, it has to be preprocessed to consolidate the most important features for vulnerability scoring and prioritization. VULCON ingests the monthly Nessus data and measures: (1) the number of months a unique vulnerability persisted on a unique host, port, and protocol combination; (2) the given vulnerability's age (measured based on CVE code, which was explained in deeper detail in Section 3.1); and (3) vulnerability severity, which is collected from the CVSS column of the Nessus output.

Table 1. A Sample of What the Master Dataset Looks Like

Obs.	CVE Code	Host	Hr. LB	Hr. UB	Hr. Rand	Severity	Age	Persistence
1	CVE-2004-8219	A	1	3	1	10	13	1
2	CVE-2012-4356	A	3	6	5	5	2	1
3	CVE-2012-4356	B	3	6	3	4	5	2
4	CVE-2009-6114	B	1	3	2	6	8	9
5	CVE-2009-6114	C	6	9	7	3	8	6
6	CVE-2004-8219	C	1	3	3	2	13	13
7	CVE-2012-4356	C	3	6	6	5	2	1
8	CVE-2009-6114	D	6	9	9	7	8	2
9	CVE-2012-4356	D	3	6	5	10	5	7

The first column ("Obs.") represents the observation number. The column labeled "CVE Code" represents the CVE code for each vulnerability. This CVE code is the value by which the year is obtained (i.e., "CVE-2004-8219" indicates the vulnerability was discovered in the year 2004 and is 13 years old as of the year 2017.) The columns labeled "Hr. LB" and "Hr. UB" represent the lower bound ("LB") and upper bound ("UB") for the personnel-hour ("Hr.") estimations. The column labeled "Hr. Rand" represents the randomly selected value within the range provided by the personnel-hour upper ("Hr. UB") and lower bounds ("Hr. LB").

In addition, it may be necessary to preprocess the scan data to do the following:

—Associate a scan's IP addresses with actual machines and network interfaces when addresses are changing due to network reconfigurations or dynamic addressing. Vendors already propose a variety of techniques and products for addressing such issues.[5]
—Filter out vulnerability instances that should not be included in the VULCON analysis. Such instances could arise when, for example, networks are segmented to isolate VIs when patches are not available or cannot be installed due to service disruption concerns. In such cases, the vulnerability scan will find those instances but the CSOC might want to knowingly remove those instances because they will not be mitigated in any case.
—Customize the analysis in other site-specific ways.

5 EXPERIMENTAL METHODOLOGY

This section provides an overview of the experimental methodology used in VULCON. The methods of obtaining, preprocessing, and consolidating the data sources are first discussed, followed by a discussion of the experimental setup for VULCON's optimization engine and, finally, trade-off analyses and comparisons across results. How the parameters were set in the model is also described.

5.1 Methodology for Merging, Consolidating, and Handling the Data

The first step in VULCON's methodology was to merge, consolidate, and process the data. A detailed overview of how this was achieved is outlined below:

(1) Merge the internal list of mission-critical services to the Nessus scan output by IP address. Add a column to the data that represents the list of mission-critical services. Populate the column with a value of 1.00 for each row that indicates the host is considered a mission-critical service and a value of 0.85 (15% penalty) in each row that indicates the host is not considered a mission-critical service.

[5] https://www.tenable.com/blog/keeping-track-of-your-ethernet-addresses.

(2) Create a unique vulnerability *key-value* pair for each month's scan output. Define the *key* as the unique appearance of a given vulnerability on a unique host, port, and protocol combination, and the *value* as a binary representation of whether or not the vulnerability appeared on that unique location on the network (host-port protocol).
(3) Iterate through the rows of the columns for severity, persistence, and age, and normalize the values so they are all on a scale between 0 and 1.
(4) Calculate the utility score as detailed in Section 3.1.1.
(5) Rank and order every unique observation of the occurrence of a vulnerability on a network location (as defined by the host-port-protocol key) according to the utility score. The vulnerability occurrences with the highest utility score arrive at the top of the queue and those with the lowest utility score arrive at the bottom of the queue.

5.2 Methodology for Selecting Parameter Values as Experimental Input

The second step is to select numerical values for the model's parameters. Experiments were run with seven different values for the personnel-hour parameter (H), starting with a lower bound of 40 and moving up in value by increments of 20 until hitting an upper bound of 160. The experiment started at 40 personnel-hours because based on subject-matter expert interviews, the best estimation is that the organization allocates an overall average of 40 personnel-hours per month to vulnerability maintenance and patching policies. The total number of vulnerabilities per scan (V) depended on the month under investigation. The smallest number of vulnerabilities in one month's scan was 126 (month 2), and the largest number of vulnerabilities in one scan was 408 (month 8).

An upper-bound limit of 6 months was set as the maximum number of months a vulnerability can wait to be selected for mitigation ($R_i = 6$). Finally, for the tradeoff analysis on the weights for the deviations on the constraints, the target TVE score was set to 100. The score of 100 was selected because the maximum TVE score for the baseline data was almost 200. Therefore, after interviewing engineers and managers on the security team, a target TVE threshold set to half that of the maximum value of the baseline result is a highly desirable improvement for any CSOC.

6 EXPERIMENT AND ANALYSIS OF RESULTS

The results of the study and experiments are described herein, including how the experimental outcome measure changes with different model inputs.

VULCON experiments were conducted on the finalized data in a manner that measured outcome changes with different model inputs. To understand VULCON's improvement against the baseline results, the first subsection compares the 40-hour baseline TVE results against the 40-hour optimized results. The second subsection compares changes in the TVE scores by changing the personnel-hour constraint (H) in increments of 20 hours, starting with a lower bound of 40 hours and an upper bound of 160 hours. The third subsection assesses changes in the parameters for the MU score by comparing the results of the 40-hour baseline performance to the 40-hour optimized performance. The fourth subsection assesses a tradeoff analysis between the weights on the individual parameters for the goal programming methodology.

The following sections include:

(1) A comparison of the TVE scores for 40-hour baseline performance against the 40-hour optimized performance (Section 6.1).
(2) A comparison of TVE scores across seven different experiments on the personnel-hour constraint (H), with changes in increments of 20 (Section 6.2).

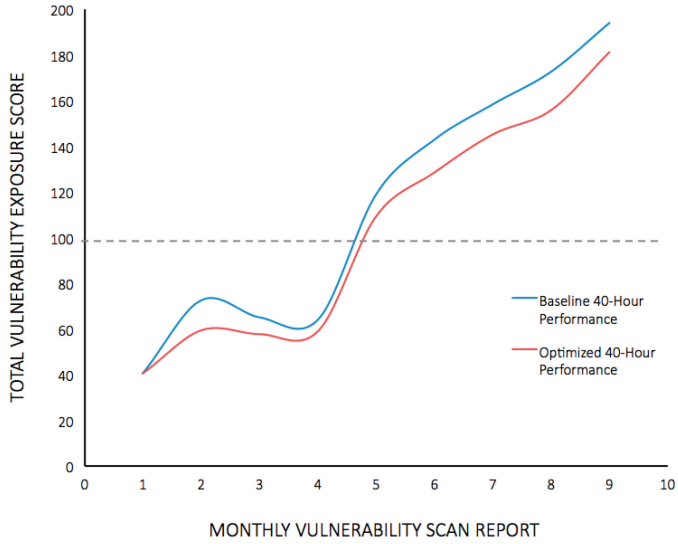

Fig. 3. The 40-hour baseline performance level compared to the 40-hour optimized performance level.

(3) An assessment of the changes in parameters of the mitigation utility score for the 40-hour baseline performance against the 40-hour optimized performance (Section 6.3).
(4) An analysis between the tradeoffs of the weights on the parameters of the goal programming model (Section 6.4).
(5) A discussion on the scalability of VULCON's experimental results.

6.1 Comparison of TVE Scores for 40-Hour Baseline Performance against 40-Hour Optimized

This section summarizes VULCON's TVE experimental results between the 40-hour baseline performance and the 40-hour optimized performance. The personnel-hours, H, and TVR value, R_i, are fixed by assigning a very large weight (10^8) to the deviation variables, d_1 and d_2. The target TVE score, S, is set to 0, and that goal is allowed to be violated by assigning a very small weight (0) to the deviation variable, d_3.

The "apple-to-apples" comparison of the results for the 40-hour baseline TVE score to that of the 40-hour optimized TVE score is provided in Figure 3. The current baseline personnel-hour performance level estimated for the CSOC is roughly 40 hours per month. The vulnerabilities for both the baseline performance and the optimized performance increase gradually from months 1 to 2 and then decrease gradually up to month 4. Month 4 introduced an onset of new vulnerabilities that is almost fives times larger than the previous month, effectively catapulting the TVE score to over 100 for both the baseline and optimized performance. The distribution of new vulnerabilities is displayed in Figure 4 to guide the reader between the volume of incoming vulnerabilities and compare it to Figure 3, displaying the trends/changes in vulnerability exposure over time.

An overall aggregate improvement of 8.97% occurred between the 40-hour baseline TVE score and the 40-hour optimized TVE scores. The maximum value for the baseline TVE score almost hit 200, while the maximum value for the optimized TVE score just barely crept up over 180.

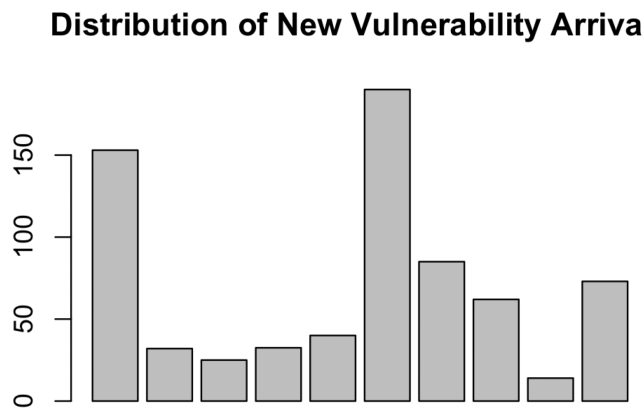

Fig. 4. The distribution of new vulnerability arrivals per month.

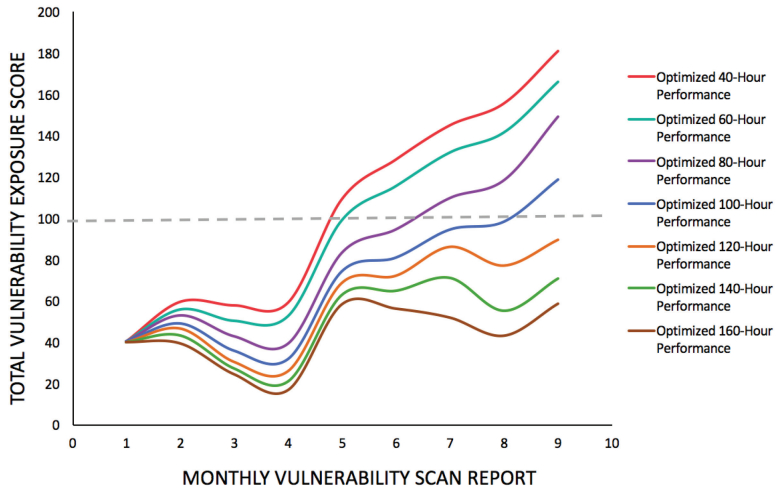

Fig. 5. A comparison of all results from the optimization output between the range of performance levels from 40 to 160 available personnel-hours.

6.2 Comparison of TVE Scores by Changing Personnel-Hour Values

This section summarizes VULCON's TVE experimental results across many different personnel-hour (H) values. The experiments provide personnel-hour inputs by increments of 20, with a range starting at 40 personnel-hours and stopping at 160 personnel-hours.

Figure 5 compares the TVE results for the complete range of optimized performance levels and shows that the overall meta-trends are similar between each personnel-hour constraint.

It was not until the personnel-hour constraint was set to 80 hours that month 5 stayed below a TVE score of 100. Additionally, the TVE score did not remain below 100 throughout the entire set of data (up to month 9) until the personnel-hour constraint was set to 120 hours. Three out of

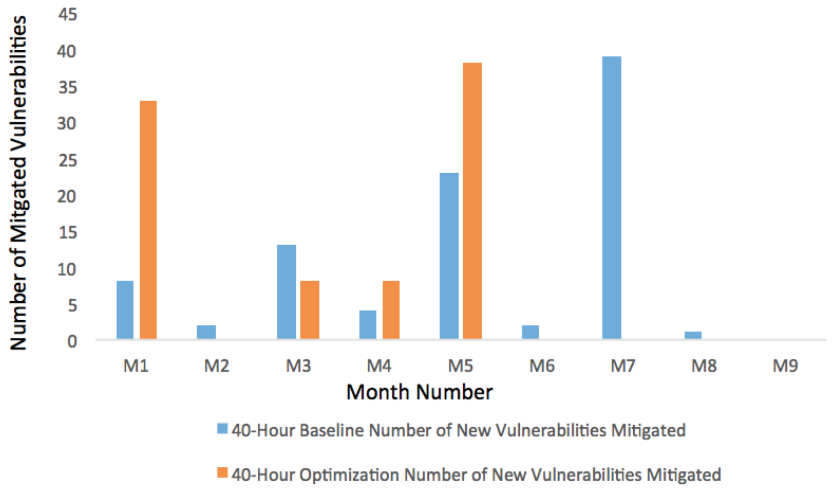

Fig. 6. A bar chart representing the number of VIs mitigated for the 40-hour baseline performance (blue) and the 40-hour optimized performance (orange) for all new (incoming) vulnerabilities per month.

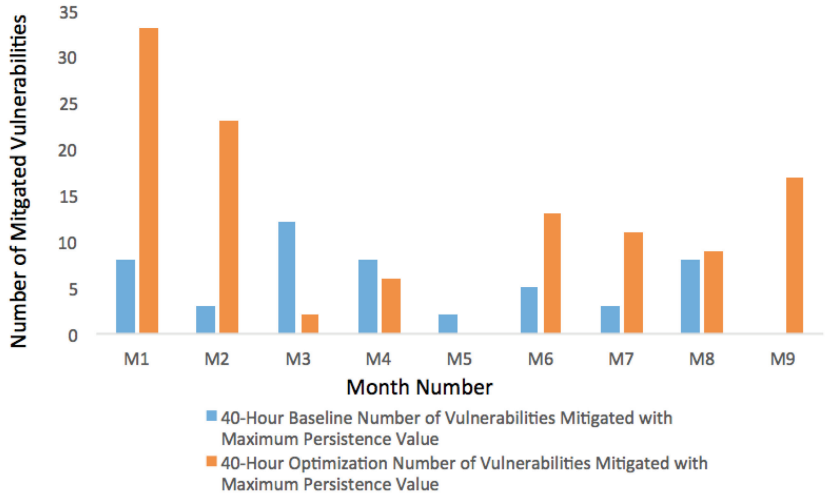

Fig. 7. A bar chart representing the number of VIs mitigated for the 40-hour baseline performance (blue) and the 40-hour optimized performance (orange) for the maximum persistence value per month.

the seven experiments continuously remained below the target TVE score of 100 (see all values for $H \geq 120$).

6.3 Assessment of Parameter Changes of MU Score When Comparing 40-Hour Baseline to 40-Hour Optimized Performance

This section reviews the changes in parameter values between the 40-hour baseline results and the 40-hour optimized results. First, the total number of vulnerabilities mitigated are reviewed in Figure 6. Next, because persistence values (p_i) are one key component VULCON aims to minimize, the maximum persistence scores are investigated in Figure 7. Next, vulnerability remediations are investigated by their most extreme severity levels (i.e., "high severity" and "critical severity") in

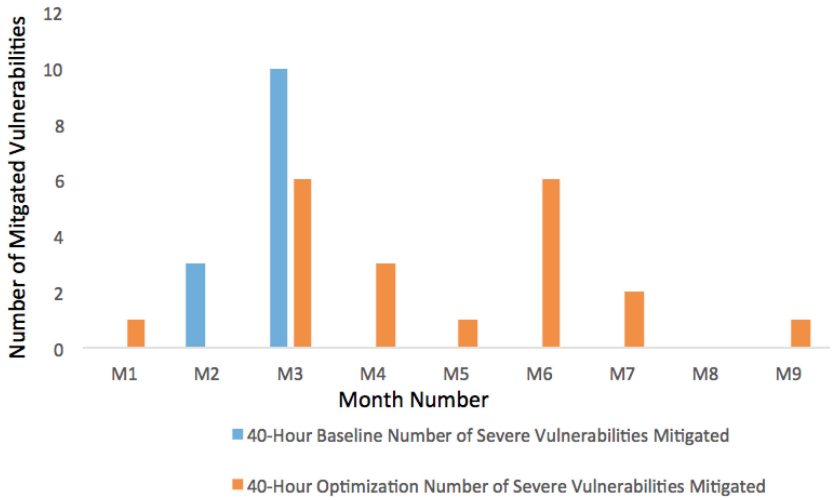

Fig. 8. A bar chart representing the number of VIs mitigated for the baseline performance (blue) and the optimized performance (orange) for the VIs with the most severe scores, that is, those categorized as "critical severity" and "high severity" by the Nessus scan output, which have associated CVSS scores that account for the *severity* (s_i) value in the utility score calculation.

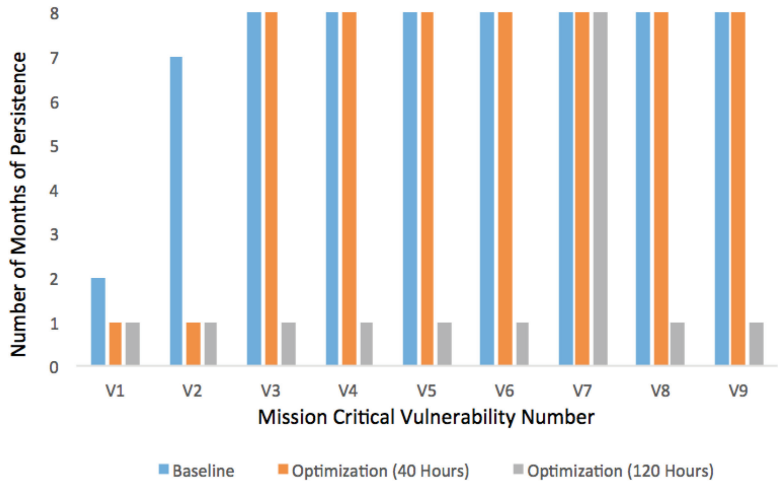

Fig. 9. A bar chart demonstrating the TVR for a vulnerability that appears on a mission-critical service.

Figure 8. Finally, the overall trends in mission-critical vulnerability remediations are represented in Figure 9.

Figure 6 demonstrates that the overall number of mitigations for the set of new vulnerabilities is slightly higher overall in the 40-hour baseline data output as compared to the 40-hour optimized output. This may not come as a surprise as the algorithm targeted vulnerabilities as an optimization between their utility scores and the number of personnel-hours required for each individual mitigation. Therefore, upon further inspection in Figure 7, the algorithm's true impact is demonstrated as the 40-hour optimization output substantially outperforms the 40-hour baseline results through mitigating more vulnerabilities with maximum persistence levels, thereby reducing the overall numbers of residual vulnerabilities that build up over time on the network.

Figure 8 demonstrates the differences in mitigation levels per month for the set of "critical severity" and "high severity" vulnerabilities, which have an associated CVSS severity score that the algorithm took into account in its utility calculation. The blue bars represent the 40-hour baseline data performance levels and the orange bars represent the 40-hour optimized performance level results. The output from the optimization algorithm show more effective mitigations for the most severe vulnerability cases of the class "critical severity" and "high severity," while the baseline output only demonstrates successful mitigations for the most severe vulnerabilities in months 1 and 2.

Figures 6, 7, and 8 demonstrate overall more mitigations of new vulnerabilities for the 40-hour baseline output than the 40-hour optimized output. However, upon further inspection, Figures 7 and 8 show that the 40-hour optimized results outperform the 40-hour baseline results in terms of mitigating the vulnerabilities that not only persisted the longest but also are the most severe (i.e., VIs with CVSS scores that correlate with the "critical" and "high" severity categories).

The algorithm and methodologies underlying VULCON provide a novel and effective vulnerability data pipeline that prioritizes VIs by taking into account not only key features (as described in Section 3.1.1) but also the personnel constraints. As can be understood from the graphs in Figures 7 and 8, higher mitigation levels occurred for VIs with maximum persistence scores in the 40-hour optimized analysis (114 total), and lower levels of vulnerability mitigations with maximum persistence scores occurred in the 40-hour baseline analysis (49 total). Additionally, there are significantly more mitigations for VIs with high severity scores. For instance, the 40-hour baseline optimization mitigated a total of 13 out of a possible 58 "critical severity" and "high severity" VIs, whereas the optimized performance mitigated 20 out of 58 (constituting a 12% improvement). As such, VULCON improves the ability to mitigate a greater number of severe VIs under the same amount of personnel-hours when comparing the baseline to optimized performances.

Figure 9 displays the effectiveness of mitigations for the VIs that appeared on mission-critical services by comparing the 40-hour baseline output, 40-hour optimized output, and 120-hour optimized output. A total of nine distinct VIs appeared on mission-critical services. The x-axis represents these nine VIs and the y-axis represents the number of months a VI persisted on the set of mission-critical services. Notice that two of the mission-critical VIs, namely, "V1" and "V2," are mitigated in the 40-hour baseline performance, whereby it took 2 months until "V1" was mitigated, and 7 months until "V2" was mitigated. Conversely, the 40-hour optimized analysis reported mitigated vulnerabilities "V1" and "V2" immediately within the first month, because at that point in time, all vulnerabilities that appeared on mission-critical services still had low persistence scores. However, as the persistence scores continued to creep up for all VIs through each month's iteration, non-mission-critical vulnerabilities started to take on more "urgency" and higher MU (u_i) scores (due to increased persistence values), which the optimization algorithm was more sensitive to. If vulnerabilities that appear on mission-critical services are a key priority for the CSOC, then, for these particular experiments, personnel-hours need to increase to at least 120 to mitigate eight out of the nine VIs.

If the CSOC wanted more mission-critical vulnerabilities to be mitigated, one of three decisions could be executed:

- Impose a steeper penalty score for non-mission-critical services in the MU value calculation (described in Section 3.1.1) so that there is a larger difference in the MU scores for mission-critical services versus non-mission-critical services.
- Increase personnel-hours so that the vulnerabilities violating the TVR constraint are mitigated and the backlog is cleared by the end of the month (i.e., the example displayed with the 120 personnel-hour results in Figure 9).

Table 2. A Representation of the Distribution of Vulnerability Remediations by Severity, Persistence, and Age for the 40-Hour Baseline Output for Both Mission-Critical Services Only and All Services

	Mission-Critical Services				All Services			
	40-Hour Baseline	Max-imum	Min-imum	Standard Deviation	40-Hour Baseline	Max. imum	Min. imum	Standard Deviation
Persistence	4.97	1	9	0.43	5.83	1	9	0.45
Severity	6.68	1	10	0.25	6.21	1	10	1.50
Age	9.08	1	17	3.97	9.15	1	17	4.56

Table 3. A Representation of the Distribution of Vulnerability Remediations by Severity, Persistence, and Age for the 40-Hour Optimized Output for Both Mission-Critical Services Only and All Services

	Mission-Critical Services				All Services			
	40-Hour Optimized	Max-imum	Min-imum	Standard Deviation	40-Hour Optimized	Max. imum	Min. imum	Standard Deviation
Persistence	2.15	1	9	1.31	3.10	1	9	0.975
Severity	5.54	1	10	1.73	5.69	1	10	3.20
Age	4.94	1	17	7.25	9.15	1	17	7.11

—Impose a new TVR constraint that is unique to the set of vulnerabilities that appear on mission-critical services so that they will remain higher in the list for selection and be mitigated within a shorter period of time.

Finally, the changes between VIs for the average persistence, severity score, and age for mission-critical services, as well as all services combined, is investigated. Table 2 represents the output for the 40-hour baseline data, and Table 3 represents the output for the optimized data. The differences between the average VIs for persistence, severity, and age for the 40-hour baseline output is compared to the 40-hour optimized output. Results are reported for mission-critical host services only and all host services (i.e., both mission critical and non-mission critical).

In addition to the changes shown for the average persistence, severity score, and age arising in a VI, the total number of mitigated VIs was 203 in the baseline performance and 192 for the optimized performance. This may occur due to the following:

—The optimization model placed more emphasis on the true "threat" of a VI by accounting for whether or not the host was on a mission-critical service (i.e., mission impact), as well as the VI's severity score, persistence, and age, rather than just sheer volume (count) of remediations. Therefore, although VULCON did not address a higher vulnerability count than the baseline, it still reduced the overall TVE score, which creates a greater impact on a CSOC's security posture.

—While the original estimation is that the baseline data represent 40 personnel-hours per month, an operational CSOC may have devoted more than 40 personnel-hours on some months, which was not necessarily tracked. This serves as evidence to some of the challenges of working with datasets from operational CSOC environments, as there could be some inconsistencies that are difficult to measure in the baseline data. This issue is further discussed in Section 7.

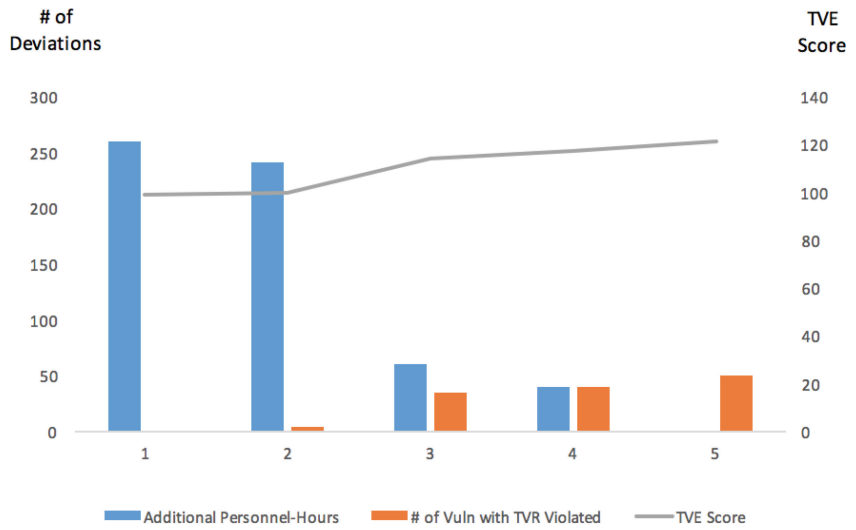

Fig. 10. A comparison of sensitivity analysis tradeoffs. The left-hand-side y-axis represents the range of deviations from either the personnel-hour or TVR constraint. The right-hand-side y-axis represents the range of TVE scores. The x-axis represents the range of solutions, where each solution is the tradeoff between the blue and orange bars. The gray TVE score line represents changes in the output TVE score from solution to solution.

The next section reviews the results for the tradeoff analysis on the weights for deviations of the constraints by perturbing the parameters of the model.

6.4 Tradeoff Analysis on the Weights for Deviations on the Constraints

This section summarizes experiments performed to understand the tradeoffs on the weights for deviations on the constraints for the VULCON model. The dataset from month 6 was used for the sensitivity experiments described herein. It is to be noted that month 6 was the iteration in which the analysis encountered a data file where the TVR value (R_i) and persistence score (p_i) were equal. Therefore, this dataset is used to demonstrate the impact of vulnerabilities with persistence values that reached the 6-month target (R_i). Based on the weights assigned to the deviations of the goals, a set of solutions could be obtained.

In Figure 10, solution 5 corresponds to the case where no additional personnel-hours (deviations on that goal) were allowed. A large weight ($w_1 = 10^8$) was assigned to the deviation variable of the personnel-hours goal, (d_1), so that it would not violate the constraint. The target TVE score (S) is defined to a value of 100, the available personnel-hours (H) to 40, and the target TVR value (R_i) to 6 months. The weight, w_3, on the deviation variable of the TVE goal (d_3) was set higher than the weight, w_2, of the deviation of the TVR goal, (d_2). As a result, the optimal solution (solution 5), selected only 13 of the 63 vulnerabilities that had a persistence value of 6 months (i.e., where $p_i = R_i = 6$), with the available personnel-hours of 40. Hence, 50 vulnerabilities, where $p_i = R_i$, were not selected for remediation. The TVE score for the remaining vulnerabilities was 120.96.

To study the interdependencies between resources (personnel-hours) and performance parameters (TVE and TVR), the weight, w_1, on the deviation d_1 was reduced so that some deviations were allowed on the personnel-hours goal. The weight, w_2, on the deviation d_2 was increased compared to the earlier case, signifying that the deviation on the TVR goal should be further minimized while the weight, w_3, on the deviation d_3 was kept the same. The output for the new set of weights

Table 4. The Range of Final Solutions for the Sensitivity Analysis of Tradeoffs Represented by the Solution Number and Corresponding Tradeoff between Additional Personnel-Hours Needed, TVR Violations, and TVE Violations

Solution Number	1	2	3	4	5
Additional Personnel-Hours	260	240	60	40	0
Number of Vulnerabilities that Violated the TVR Constraint	0	3	34	39	50
TVE Score	98.852	100	114.206	116.791	120.958

is shown in solution 4 of Figure 10. The TVE score was further reduced to 116.79 by utilizing an additional 40 personnel-hours. The number of vulnerabilities that violated the target TVR value of 6, d_2, was reduced from 50 to 39.

A further reduction on the weight, w_1, on the deviation d_1 produced an output shown as solution 3 in Figure 10. With a higher number of deviations on the target personnel-hours ($d_1 = 60$), the TVE score was reduced to 114.21. The number of vulnerabilities that violated/deviated from the target TVR value of 6, d_2, was reduced to 34.

Solution 2 in Figure 10 corresponds to the case where there were no deviations on the TVE goal (i.e., the TVE score of the remaining vulnerabilities met the target of 100). To satisfy this goal, the weight, w_1, on the deviation d_1 was significantly reduced, while the weight, w_2, on the deviation d_2 was set higher than the previous case. As a result, only three vulnerabilities remained that violated the TVR goal, while the rest of them (where $p_i = R_i$) were selected for remediation with an additional 240 personnel-hours ($d_1 = 240$) requirement.

Finally, solution 1 in Figure 10 represents the case where no deviations in either of the performance goals (TVE and TVR) occurred. The weight, w_2, on the deviation d_2 was set even higher than in the previous case such that no vulnerabilities remained that violated the TVR goal. All the vulnerabilities where $p_i = R_i$ were selected for remediation. The additional personnel-hours ($d_1 = 260$) required were 260, which resulted in a TVE score of 98.85. Table 4 represents the range of final solutions, with their respective tradeoffs.

As shown above, an organization can obtain various solutions using the goal programming method to perform a comparative analysis between the various parameters and obtain important insights on their interdependencies. The knowledge gained from this model can help an organization better manage and execute vulnerability mitigation strategies.

Finally, the remaining sections investigate issues surrounding scalability and challenges to internal and external validity, and summarize the meta-principles from VULCON's experimental results.

6.5 Scalability

While investigating a few vulnerabilities on a few hosts is simple, such an approach will not scale up to large, enterprise networks. When a CSOC has to assess hundreds of thousands of vulnerabilities, spanning hundreds of thousands of hosts, the complexity increases exponentially. The experimental results described in this article provide further evidence and support for VULCON's greater impact on managing and prioritizing vulnerabilities. To build on and extend this work, the following areas would improve the scalability capacity for VULCON:

(1) Structured data tracking programs (e.g., CSOCs routinely keeping the list of mission-critical services up to date).

(2) Computational complexity differences in optimization programs for extremely large enterprise networks.
(3) Organizational disruptions in personnel management, allocation of responsibilities, and assigning ownership to which host services need to be patched.

With the right management routines and organizational commitment to a structured vulnerability response program, these scalability challenges can be investigated and overcome.

7 CHALLENGES TO INTERNAL AND EXTERNAL VALIDITY

This section reviews identified challenges to internal and external validity. Recognizing and mitigating these challenges are critical to both generating and processing the data that produce reliable results and thorough scientific analysis. Yet, there are some challenges to the internal and external validity in the VULCON experiments.

7.1 Internal Validity

Internal validity refers to how well an experiment was conducted. Factors that support good internal validity are having as few confounding variables as possible. As the number of independent variables acting at the same time increases, so do the challenges to internal validity. Some of the challenges to internal validity for VULCON's experiments include servers being taken offline during data collection, scan policy changes, a 1-month gap in the dataset, and potential software flaws in the vulnerability scanner such as false positives.

Servers can temporarily be taken offline or decommissioned for various reasons. It may be the case that a system has to be upgraded or access policies change. Temporary safeguards may need to be imposed if a "bug" is found on a database. If so, the server is taken offline to prevent further damage until it can be investigated later. Additionally, scan policies can change over time. When an enterprise-level network undergoes periodic changes, such as permanently adding and deleting servers, host names need to be either added or deleted from the organization's current scan policy.

Finally, software flaws are problems in the software of the Nessus Scanner itself and can contribute to problems such as false positives. An existing body of research aims to address vulnerability software false positives and false negatives. The curious reader is directed to the work of Holm et al. (2011, 2012) and Doupé et al. (2012) to learn more.

7.2 External Validity

External validity refers to how well scientific results generalize to similar circumstances. The less a scientific study generalizes to other studies, the greater the challenges are to external validity. The primary challenge to external validity is that the VULCON experiments are effectively observational studies from one organization. It is ideal to have data from multiple and distinct organizations to improve the significance of VULCON's experimental observations.

An additional challenge to external validity is that this analysis used data that are solely from the defender's side of cyber-security. As it fell beyond the scope of this study, VULCON did not take exploitation data into account, therefore constituting a challenge to external validity. Indeed, optimized vulnerability management systems that take exploitation data into account provide ample opportunity for future work.

8 CONCLUSION, SUMMARY OF META-PRINCIPLES, AND DISCUSSION OF FUTURE WORK

The objective of this article was to propose a vulnerability management strategy called VULCON (VULnerability CONtrol) for cyber-security operating centers (CSOCs). A principled analysis of

CSOC vulnerability management and prioritization was reviewed through the introduction and experimentation on the VULCON framework. Additionally, two new evaluation metrics were defined to improve measurements in CSOC performance: (1) total vulnerability exposure (TVE) and (2) time-to-vulnerability remediation (TVR). A data pipeline and preprocessing algorithm was developed to quantify the mitigated utility (MU) scores of the occurrence on vulnerabilities on unique hosts for more principled vulnerability prioritization. A multiobjective optimization model was then used to determine which vulnerabilities can be patched first, based on total available personnel-hours and TVE and TVR requirements. VULCON outputs metrics to include updated TVE scores based on the monthly vulnerability process, the list of patched/remediated vulnerabilities, and the list of unremediated (residual) vulnerabilities. The set of residual vulnerabilities was fed back into the VULCON algorithm for the following month, and the persistence and utility scores were updated as the algorithm iterated through each month. In assessing the final results, there was an overall improvement of 8.97% in the TVE score between the 40-hour optimized results and the baseline results. When comparing the optimized outputs from the TVR constraints of 40 personnel-hours and up to 160 personnel-hours, in increments of 20, the TVE score did not fall below a designated threshold of 100 until the personnel requirement hit 120 hours. A sensitivity analysis was conducted to evaluate the tradeoffs between the model's parameters on the 40-hour optimization output. In order to keep the TVE score continuously below 100, the model required a total of 260 personnel-hours by the end of month 6, concluding at a TVE score of 98.85. In conclusion, the results of the VULCON experiments have shown that organizations can obtain many different types of solutions using the multiobjective optimization method.

8.1 Summary of Meta-Principles

The meta-principles concluded from this study are that vulnerability management programs are more powerful if they account for many different features to include a vulnerability's age, persistence, and severity score, and whether or not it appears on a mission-critical service. Together, these features create a "score" that can be used to prioritize vulnerabilities. Such an approach to vulnerability management has far more impact on the network's security posture than simply focusing on volume (count) of vulnerability remediations. Additionally, a core principle of this study aims at taking into account the number of personnel-hours required per vulnerability remediation against the total available personnel-hours of the organization as a whole. By processing vulnerabilities in such a fashion, this is a more effective strategy to reduce overall vulnerability exposure.

8.2 Discussion of Future Work

Areas of opportunity for future work consist of computational scalability, developing methods for implementing structured vulnerability response programs in CSOCs, addressing false-positive rates in the Nessus output, and analyzing how the incorporation of more granular information on vulnerability age (e.g., accounting for age in terms of months, or days, rather than years) could provide more precise estimates of the TVE scores. Investigations into the impact VULCON could have on reducing analyst burnout and the correlation between vulnerabilities (i.e., could more time be saved if correlated vulnerabilities can be fixed at the same time?) would also provide meaningful future work. Additionally, the areas of stochastic optimization and reinforcement learning would serve as good follow-on work to experiment with the precise allocation and forecasting of the number of personnel-hours to dynamically allocate and schedule personnel. See the work of Ganesan et al. (2016, 2017) and Shah et al. (2017) for examples of analyst scheduling and reinforcement learning as it applies to a CSOC's level of operational effectiveness (LoE). Similar types of analysis could be applied in this area. Finally, understanding the unintended adverse effects of

organizational metrics, such as the TVE and TVR scores, on employee behaviors and performance incentives is a worthy area of investigation.

ACKNOWLEDGMENTS

The authors would like to thank the CSOC team for providing their vulnerability data for this analysis.

REFERENCES

2017. *Microsoft Exploitability Index*. Accessed September 29, 2017. Retrieved from https://technet.microsoft.com/en-us/security/cc998259.

2017. *Symantec Threat Severity Assessment*. Accessed September 29, 2017. Retrieved from https://www.symantec.com/security_response/severityassessment.jsp.

2017. *Tenable Network Security*. Accessed September 29, 2017. Retrieved from https://www.tenable.com/blog/new-nessus-feature-added-csv-export.

2018. *Creating a Patch and Vulnerability Management Program, Recommendations of the National Institute of Standards and Technology (NIST)*. Accessed March 21, 2018. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-40ver2.pdf.

2018. *Payment Card Industry (PCI) Data Security Standard*. Accessed March 21, 2018. Retrieved from https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2.pdf?agreement=true&time=1521778935419.

Anthony Afful-Dadzie and Theodore T. Allen. 2014. Data-driven cyber-vulnerability maintenance policies. *Journal of Quality Technology* 46, 3 (2014), 234.

Anthony Afful-Dadzie and Theodore T. Allen. 2016. Control charting methods for autocorrelated cyber vulnerability data. *Quality Engineering* 28, 3 (2016), 313–328.

Luca Allodi and Fabio Massacci. 2014. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security (TISSEC)* 17, 1 (2014), 1.

Belaid Aouni, Cinzia Colapinto, and Davide La Torre. 2014. Financial portfolio management through the goal programming model: Current state-of-the-art. *European Journal of Operational Research* 234, 2 (2014), 536–545.

Seyed Mohammad Asadzadeh and Ali Azadeh. 2014. An integrated systemic model for optimization of condition-based maintenance with human error. *Reliability Engineering & System Safety* 124 (2014), 117–131.

Christian Biener, Martin Eling, and Jan Hendrik Wirfs. 2015. Insurability of cyber risk: An empirical analysis. *Geneva Papers on Risk and Insurance-Issues and Practice* 40, 1 (2015), 131–158.

Jean Camp, Lorrie Cranor, Nick Feamster, Joan Feigenbaum, Stephanie Forrest, David Kotz, Wenke Lee, Patrick Lincoln, Vern Paxson, Mike Reiter, Ron Rivest, William Sanders, Stefan Savage, Sean Smith, Eugene Stafford, and Sal Stolfo. 2009. Data for cybersecurity research: Process and "Wish List". Retrieved from http://www.ljean.com/files/data-wishlist.pdf.

Hasan Cavusoglu, Huseyin Cavusoglu, and Jun Zhang. 2008. Security patch management: Share the burden or share the damage? *Management Science* 54, 4 (2008), 657–670.

Kelley Dempsey, Nirali Shah Chawla, Arnold Johnson, Ronald Johnston, Alicia Clay Jones, Angela Orebaugh, Matthew Scholl, and Kevin Stine. 2012. Information security continuous monitoring (ISCM) for federal information systems and organizations. CreateSpace Independent Publishing Platform, National Institute of Standards and Technology Special Publication 800-137.

Adam Doupé, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna. 2012. Enemy of the state: A state-aware black-box web vulnerability scanner. In *USENIX Security Symposium*. 523–538.

Tudor Dumitras and Darren Shou. 2011. Toward a standard benchmark for computer security research. In *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS Workshop)*. Citeseer.

Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2017. Optimal scheduling of cybersecurity analysts for minimizing risk. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 4 (2017), 52.

Rajesh Ganesan, Sushil Jajodia, Ankit Shah, and Hasan Cam. 2016. Dynamic scheduling of cybersecurity analysts for minimizing risk using reinforcement learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 4.

Richard J. Holden. 2011. Lean thinking in emergency departments: A critical review. *Annals of Emergency Medicine* 57, 3 (March 2011), 265–278. DOI: http://dx.doi.org/10.1016/j.annemergmed.2010.08.001

Hannes Holm, Mathias Ekstedt, and Dennis Andersson. 2012. Empirical analysis of system-level vulnerability metrics through actual attacks. *IEEE Transactions on Dependable and Secure Computing* 9, 6 (2012), 825–837.

Hannes Holm, Teodor Sommestad, Jonas Almroth, and Mats Persson. 2011. A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security* 19, 4 (2011), 231–247.

Leora I. Horwitz, Jeremy Green, and Elizabeth H. Bradley. 2010. US emergency department performance on wait time and length of visit. *Annals of Emergency Medicine* 55, 2 (February 2010), 133–41. DOI: http://dx.doi.org/10.1016/j.annemergmed.2009.07.023

James P. Ignizio. 1983. Generalized goal programming. An overview. *Computers and Operations Research* 10, 4 (1983), 277–289.

Omid Jadidi, S. Zolfaghari, and Sergio Cavalieri. 2014. A new normalized goal programming model for multi-objective problems: A case of supplier selection and order allocation. *International Journal of Production Economics* 148 (2014), 158–165.

Miles A. McQueen, Trevor A. McQueen, Wayne F. Boyer, and May R. Chaffin. 2009. Empirical estimates and observations of 0day vulnerabilities. In *42nd Hawaii International Conference on System Sciences, 2009 (HICSS'09)*. IEEE, 1–12.

Peter Mell, Karen Scarfone, and Sasha Romanosky. 2007. A complete guide to the common vulnerability scoring system version 2.0. In *FIRST-Forum of Incident Response and Security Teams*. 1–23.

Christos H. Papadimitriou. 1981. On the complexity of integer programming. *Journal of the ACM (JACM)* 28, 4 (1981), 765–768.

Ronald L. Rardin. 1998. *Optimization in Operations Research*. Prentice-Hall.

Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2017. A methodology to measure and monitor level of operational effectiveness of a CSOC. *International Journal of Information Security* 17 (2017), 1–14.

Shari J. Welch, Brent R. Asplin, Suzanne Stone-Griffith, Steven J. Davidson, James Augustine, Jeremiah Schuur, and Emergency Department Benchmarking Alliance. 2011. Emergency department operational metrics, measures and definitions: Results of the Second Performance Measures and Benchmarking Summit. *Annals of Emergency Medicine* 58, 1 (July 2011), 33–40. DOI: http://dx.doi.org/10.1016/j.annemergmed.2010.08.040

Received May 2017; revised November 2017; accepted March 2018