Verified Metrics for Continuous Active Defence

George O. M. Yee
Computer Research Lab, Aptusinnova Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
email: george@aptusinnova.com, gmyee@sce.carleton.ca

***Abstract*—As a sign of the times, headlines today are full of attacks against an organization's computing infrastructure, resulting in the theft of sensitive data. In response, the organization applies security measures (e.g., encryption) to secure its vulnerabilities. However, these measures are often only applied once, with the assumption that the organization is then protected and no further action is needed. Unfortunately, attackers continuously probe for vulnerabilities and change their attacks accordingly. This means that an organization must also continuously check for new vulnerabilities and secure them, to continuously and actively defend against the attacks. This paper derives metrics that characterize the security level of an organization at any point in time, based on the number of vulnerabilities secured and the effectiveness of the securing measures. The metrics are verified in terms of their soundness using the author's recently published procedure for deriving good security metrics. The paper then shows how an organization can apply the metrics for continuous active defence.**

Keywords- sensitive data; vulnerability; security level; verified metrics; continuous defence.

I. Introduction

This work extends Yee [1] by adding explanations and related work, elaborating the application areas, and including a new section on verifying the soundness of the proposed metrics.

Headlines today are full of news of attacks against computing infrastructure, resulting in sensitive data being compromised. These attacks have devastated the victim organizations. The losses have not only been financial (e.g., theft of credit card information), but perhaps more importantly, have damaged the organizations' reputation. The first half of 2019 had 3,800 publicly disclosed breaches with 4.1 billion records exposed, an increase of 54% in the number of reported breaches when compared to the first half of 2018 [2]. Here are just two of those breaches [2]:

- March 22 and 23, 2019, Capital One: The number of records breached was 106 million, including names, addresses, postal codes, phone numbers, email addresses, birthdates, and self-reported income. Also exposed in some cases were customer credit scores, credit limits, balances, and payment history. The breach affected about 100 million consumers in the United States and about 6 million in Canada. A hacker accessed the servers of a third-party cloud services company contracted by Capital One. The hacker hacked the servers on March 22 and 23, 2019 and has since been arrested. According to CNN Business, Capital One expected losses of $100 million to $150 million related to the hack, for expenses incurred in notifying affected customers, providing free credit monitoring, legal defense, and fixing the vulnerability.
- August 1, 2018 to March 30, 2019, American Medical Collection Agency: Here the number of records breached was over 20 million, including social security numbers, birthdates, payment card data, credit card information, and bank account information. American Medical Collection Agency collected overdue payments for medical labs. This long running breach exposed the records of the labs' customers including the above sensitive data. A cybersecurity firm found the breached information on the dark web. American Medical Collection Agency filed for bankruptcy in June 2019, citing IT costs, possible lawsuits, and the loss of business from its customers.

Hard hit data breach victims in 2018 [3] include toymaker Vtech Technologies (a cyberattack exposed the personal data of an estimated 6.4 million children worldwide), Under Armour (a cyberattack stole the personal data of 150 million users of its app), and major airlines such as Air Canada, British Airways, and Cathay Pacific (hackers made off with the personal data of a combined 9.8 million customers). The year 2017 [4] saw a total of 5,207 breaches and 7.89 billion information records compromised.

In response to attacks, such as the ones described above, organizations determine their computer system vulnerabilities and secure them using security measures. Typical measures include firewalls, intrusion detection systems, two-factor authentication, encryption, and training for employees on identifying and resisting social engineering. However, once the security measures have been implemented, organizations tend to believe that they are safe and that no further actions are needed. Unfortunately, attackers do not give up just because the organization has secured its known computer vulnerabilities. Rather, the attackers will continuously probe the organization's computer system for new vulnerabilities that they can exploit. This means that the organization must continuously analyze its computer system vulnerabilities and secure any new ones that it discovers. In order to do this effectively, it is useful to have quantitative metrics of the security level at any particular point in time, based on the number of vulnerabilities secured and the effectiveness of the security measures, at that point in time. An acceptable security level can be set, so that if the security level falls below this acceptable level due to new vulnerabilities, the latter can be secured to bring the security level back to the acceptable level. This work derives such metrics and shows how to apply them for continuous active defence, i.e., continuous vulnerabilities evaluation and follow up. Further, this work verifies that the proposed metrics are sound, a term that will be defined below.

The objectives of this work are: i) derive straightforward, clear metrics of the resultant protection level obtained by an organization at any point in time, based on the use of security measures to secure vulnerabilities and based on the effectiveness of the measures, ii) show how these metrics can be calculated, iii) verify that these metrics are sound, and iv) show how the metrics can be applied for continuous active defence and discuss some application areas. We seek straightforward, easy to understand metrics since complicated, difficult to understand ones tend not to be used or tend to be misapplied. We base these metrics on securing vulnerabilities since this has been and continues to be the method organizations use to secure their computer infrastructure.

The rest of this paper is organized as follows. Section II discusses sensitive data, attacks, and vulnerabilities. Section III derives the metrics, shows how to calculate them, and presents various aspects of the metrics, including some of their strengths, weaknesses, and limitations. Section IV verifies that the metrics are sound. Section V explains how to apply the metrics for continuous active defence and presents some application areas. Section VI discusses related work. Section VII gives conclusions and future work.

II. Sensitive Data, Attacks, and Vulnerabilities

Sensitive data is data that needs protection and must not fall into the wrong hands. It includes private or personal information [5], which is information about an individual, can identify that individual, and is owned by that individual. For example, an individual's height, weight, or credit card number can all be used to identify the individual and are considered as personal information or personal sensitive data. Sensitive data also includes non-personal information that may compromise the competitiveness of the organization if divulged, such as trade secrets or proprietary algorithms and formulas. For government organizations, non-personal sensitive data may include information that is vital for the security of the country for which the government organization is responsible.

DEFINITION 1: *Sensitive data* (SD) is information that must be protected from unauthorized access in order to safeguard the privacy of an individual, the well-being or expected operation of an organization, or the well-being or expected functioning of an entity for which the organization has responsibility.

DEFINITION 2: An *attack* is any action carried out against an organization's computer system that, if successful, compromises the system or the SD held by the system.

An attack that compromises a computer system is Distributed Denial of Service (DDoS). One that compromises the SD held by the system is a Trojan horse attack in which malicious software (the Trojan) is planted inside the system to steal SD. Attacks can come from an organization's employees, in which case the attack is an *inside attack*. For example, a disgruntled employee secretly keeps a copy of a SD backup and sells it on the "dark web".

DEFINITION 3: A *vulnerability* of a computer system is any weakness in the system that can be targeted by an attack with some expectation of success. A vulnerability can be secured to become a *secured vulnerability* through the application of a security measure.

An example of a vulnerability is a communication channel that is used to convey sensitive data in the clear. This vulnerability can be targeted by a Man-in-the-Middle attack with reasonable success of stealing the sensitive data. This vulnerability can become a secured vulnerability by encrypting the sensitive data that the communication channel carries.

A computer system can undergo upgrades, downgrades, and other modifications over time that changes its number of secured and unsecured vulnerabilities. It is thus necessary to specify a time *t* when referring to vulnerabilities. Clearly, the number of secured and unsecured vulnerabilities of a computer system at time *t* is directly related to the security level of the system at time *t*. This idea is formalized in the next definition.

DEFINITION 4: A computer system's security level (SL) at time t, or SL(t), is the degree of protection from attacks that results from having q(t) secured vulnerabilities, and p(t) unsecured vulnerabilities, where the system has a total of N(t) = p(t)+q(t) secured and unsecured vulnerabilities. SL(t) is uniquely represented by the pair (p(t), q(t)).

Clearly SL(t) increases with increasing q(t) and decreases with increasing p(t). Figure 1 shows 3 SL(t) points on the (p(t), q(t)) plane for N(t)=100.

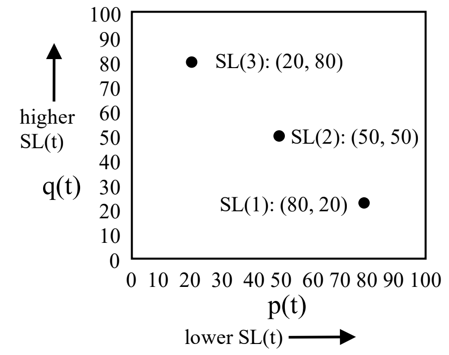

Figure 1. SL(t) points corresponding to a computer system with N(t)=100. SL(3) is higher security than SL(2), which is higher security than SL(1).

In Figure 1, the higher values of q(t) correspond to higher security levels, and the higher values of p(t) correspond to lower security levels.

Definition 4 requires p(t) to be known. Of course, it is next to impossible to determine all the vulnerabilities in a typical computer system, so the exact value of p(t) is most likely undeterminable. Thus, a value for p(t) can only be a "best effort" value, and consequently, a value for SL(t) is not the true value, but a "best effort" value. It is in this context that the values of p(t) and SL(t) are to be understood.

III. Metrics for Continuous Active Defence

While the pair (p(t), q(t)) uniquely represents SL(t), it cannot be used to calculate the value of SL(t), which would be useful in tracking the security of a system over time as its vulnerabilities change. In this section, we derive two metrics for the value of SL(t), one assuming that the measures securing vulnerabilities are totally reliable; the other with the measures only partly reliable. Both metrics are applied right after the vulnerabilities have been determined, and possibly before any of them have actually been secured. Determining vulnerabilities is discussed in Section III.C below.

A. Metric with Totally Reliable Securing Measures

We seek a metric STRM(t) (STRM is an acronym for "SL with Totally Reliable Measures") for a computer system's SL(t), where all securing measures are totally reliable. Suppose that p(t) and q(t) are as in Definition 4. Let $P_t(e)$ represent the probability of event e at time t. Let "exploit" mean a successful attack on a vulnerability. Let "all exploits" mean exploits on 1 or more vulnerabilities. Let $U_k(t)$ denote an unsecured vulnerability k at time t. We have

$$SL(t) = P_t(\text{no exploits}) = 1\text{-}P_t(\text{all exploits}) \qquad (1)$$

However, the only exploitable vulnerabilities are the unsecured vulnerabilities since the securing measures are totally reliable. Therefore

$$P_t(\text{all exploits}) = \Sigma_k\ [P_t(\text{exploit of } U_k(t))]$$

by applying the additive rule for the union of probabilities, assuming that 2 or more exploits do not occur simultaneously. Let $u_k(t)$ be a real number with $0 < u_k(t) \leq p(t)$ and $\Sigma_k u_k(t) = p(t)$. Set

$$P_t(\text{exploit of } U_k(t)) \approx u_k(t)/(p(t)+q(t)) \qquad (2)$$

By substitution using (2)

$$\begin{aligned} P_t(\text{all exploits}) &\approx \Sigma_k\ [u_k(t)/(p(t)+q(t))] \\ &= \Sigma_k u_k(t)/(p(t)+q(t)) \\ &= p(t)/(p(t)+q(t)) \qquad (3) \end{aligned}$$

The condition $0 < u_k(t) \leq p(t)$ is needed to ensure that there is some probability for an unsecured vulnerability to be exploited. The condition $\Sigma_k u_k(t) = p(t)$ is necessary in order for $P_t(\text{all exploits}) \leq 1$. Expression (2) gives a way of assigning values for $P_t(\text{exploit of } U_k(t))$ based on a risk analysis [5]. However, expression (3) ensures that such assignment is not needed for calculating STRM(t). In other words, the fact that some vulnerabilities are more likely to be exploited than others does not affect the value of STRM(t).

Substituting (3) into (1) gives

$$\begin{aligned} SL(t) &\approx 1\text{-}[p(t)/(p(t)+q(t))] \\ &= q(t)/(p(t)+q(t)) \quad \text{if } p(t)+q(t) > 0 \\ &= 1 \quad \text{if } p(t)+q(t) = 0 \end{aligned}$$

(Note that mathematically, we cannot divide by 0.) We obtain STRM(t) by assigning as follows:

$$\begin{aligned} \mathbf{STRM(t)} &\mathbf{= q(t)/(p(t)+q(t))} \quad \mathbf{if\ p(t)+q(t) > 0} \qquad (4) \\ &\mathbf{= 1} \quad \mathbf{if\ p(t)+q(t) = 0} \qquad (5) \end{aligned}$$

We see from (4) that $0 \leq STRM(t) \leq 1$ if $p(t)+q(t) > 0$ and has value 0 if q(t)=0 (the system has no secured vulnerabilities) and 1 if p(t)=0 (all of its vulnerabilities are secured). We see from (5) that STRM(t)=1 if p(t)+q(t)=0 (no vulnerabilities, which is unlikely). The values of the metric are therefore as expected.

B. Metric with Partially Reliable Securing Measures

Here, we seek a metric SPRM(t) (SPRM is an acronym for "SL with Partially Reliable Measures") for a computer system's SL(t) where the measures securing the vulnerabilities are only partially reliable.

Let $V_k(t)$ denote a secured vulnerability k at time t. The reliability $r_k(t)$ of the measure securing $V_k(t)$ can be defined as the probability that the measure remains operating from time zero to time t, given that it was operating at time zero [6]. The unreliability of the measure is then $1\text{-}r_k(t)$. We have the events

[exploit of $V_k(t)$] if and only if [$V_k(t)$ selected for exploit] AND [measure securing $V_k(t)$ unreliable]

Since the two right-hand side events are independent,

$$P_t(\text{exploit of } V_k(t)) = P_t(V_k(t) \text{ selected for exploit}) \times P_t(\text{measure securing } V_k(t) \text{ unreliable})$$

Set $\quad P_t(V_k(t) \text{ selected for exploit}) \approx 1/(p(t)+q(t)) \qquad (6)$

since attackers will have no preference to attack one secured vulnerability over another secured vulnerability (they should not even see them as vulnerabilities). Again, applying the additive rule for the union of probabilities,

$$\begin{aligned} P_t(\text{all } V_k(t) \text{ exploits}) &= \Sigma_k[P_t(V_k(t) \text{ selected for exploit}) \times P_t(\text{measure securing } V_k(t) \text{ unreliable})] \\ &= \Sigma_k\ [(1/(p(t)+q(t)))(1\text{-}r_k(t))] \\ &= [\Sigma_k(1\text{-}r_k(t)]/[p(t) + q(t)] \\ &= [q(t)\text{-}\Sigma_k r_k(t)]/[p(t) + q(t)] \\ &=[q(t)/(p(t)+q(t))]\text{-}\Sigma_k r_k(t)/(p(t) + q(t)) \qquad (7) \end{aligned}$$

Now, since both $U_k(t)$ and $V_k(t)$ can be exploited,

$$\begin{aligned} P_t(\text{all exploits}) &= P_t(\text{all } U_k(t) \text{ exploits}) + P_t(\text{all } V_k(t) \text{ exploits}) \\ &\approx [p(t)/(p(t)+q(t))] + [q(t)/(p(t)+q(t))]\text{-} \Sigma_k r_k(t)/(p(t) + q(t)) \\ &= 1 - \Sigma_k r_k(t)/(p(t) + q(t)) \qquad (8) \end{aligned}$$

by substitution using (3) and (7), where (3) is $P_t(\text{all } U_k(t)$

exploits). Finally, by substitution using (1) and (8),

$$\begin{aligned} SL(t) &\approx 1 - 1 + \Sigma_k r_k(t)/(p(t) + q(t)) \\ &= \Sigma_k r_k(t)/(p(t) + q(t)) && \text{if } p(t) \geq 0, q(t) > 0 \\ &= 1 && \text{if } p(t)+q(t) = 0 \\ &= 0 && \text{if } p(t)>0, q(t) = 0 \end{aligned}$$

We obtain SPRM(t) by assigning as follows:

$$\begin{aligned} \mathbf{SPRM(t)} &\mathbf{= \Sigma_k r_k(t)/(p(t)+q(t))} && \mathbf{if\ p(t) \geq 0,\ q(t) > 0} && (9) \\ &\mathbf{= 1} && \mathbf{if\ p(t)+q(t) = 0} && (10) \\ &\mathbf{= 0} && \mathbf{if\ p(t)>0,\ q(t)=0} && (11) \end{aligned}$$

We see from (9) that $0 < SPRM(t) < 1$ for $p(t) \geq 0$, $q(t) > 0$ (all vulnerabilities may or may not be secured), and from (10) that $SPRM(t) = 1$ for $p(t)+q(t) = 0$ (no vulnerabilities, which is unlikely). We see from (11) that $SPRM(t) = 0$ for $p(t)>0$, $q(t) = 0$ (no secured vulnerabilities). We also see that for $r_k(t) = 1$, SPRM(t) is the same as STRM(t). The values of the metric are therefore as expected.

C. *Calculating the Metrics*

Calculating STRM(t) requires the values of p(t) and q(t) at a series of time points of interest. SPRM(t) requires the values of p(t), q(t), and the reliability value for each measure used to secure the vulnerabilities.

To obtain the values of p(t) and q(t), an organization may perform a threat analysis of vulnerabilities in the organization's computer system that could allow attacks to occur. Threat analysis or threat modeling is a method for systematically assessing and documenting the security risks associated with a system (Salter et al. [7]). Threat modeling involves understanding the adversary's goals in attacking the system based on the system's assets of interest. It is predicated on that fact that an adversary cannot attack a system without a way of supplying it with data or otherwise accessing it. In addition, an adversary will only attack a system if it has some assets of interest. The method of threat analysis given in [7] or any other method of threat analysis will yield the total number N(t) of vulnerabilities to attacks at time t. Once this number is known, the organization can select which vulnerabilities to secure and which security measures to use, based on a prioritization of the vulnerabilities and the amount of budget it has to spend. A way to optimally select which vulnerabilities to secure is described in [8]. Once vulnerabilities have been selected to be secured, we have q(t). Then $p(t) = N(t) - q(t)$. The threat analysis may be carried out by a project team consisting of the system's design manager, a security and privacy analyst, and a project leader acting as facilitator. In addition to having security expertise, the analyst must also be very familiar with the organization's computer system. Further discussion on threat analysis is outside the scope of this paper. More details on threat modeling can be found in [8]. Vulnerabilities may be prioritized using the method in [5], which describes prioritizing privacy risks.

The reliability values for hardware measures used to secure the selected vulnerabilities may be obtained from the hardware's manufacturers (e.g., hardware firewall). Reliability values for software and algorithmic measures are more difficult to obtain (e.g., encryption algorithm). For these, it may be necessary to estimate the reliability values based on the rate of progress of technology. For example, one could estimate the reliability of an encryption algorithm based on estimates of the computer resources that attackers have at their disposal. If they have access to a super computer, an older encryption algorithm may not be sufficiently reliable. One could also opt to be pessimistic and assign low reliability values, which would have the net effect of boosting security by securing more vulnerabilities, in order to meet a certain SL(t) level (see Section V). Reliability values for security measures represent a topic for future research.

It is important to note that at each time point where the metrics are calculated, the values of p(t) and q(t) are generated anew. Vulnerabilities secured previously with totally reliable measures would not appear again as vulnerabilities. On the other hand, vulnerabilities secured with only partially reliable measures should be identified again as vulnerabilities. Further, it is not necessary to have actually implemented the securing measures before calculating the metrics.

D. *Graphing the Metrics*

The metrics STRM(t) and SPRM(t) are both functions of p(t), q(t), and t. Figure 2 shows a 3-dimensional graph of these metrics with axes for STRM(t)/SPRM(t), p(t), and q(t). Time is not shown explicitly as an axis since we would need 4 dimensions, but is instead represented as time period displacements of the metrics' values.

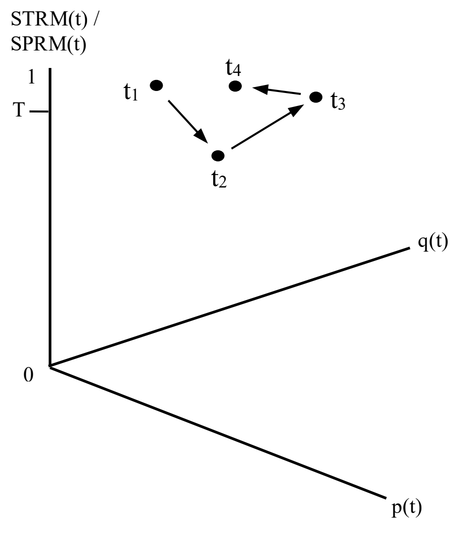

Figure 2. STRM(t)/SPRM(t) values at times $t_1 < t_2 < t_3 < t_4$.

Figure 2 shows 4 values of one of the metrics, labeled according to the times it was evaluated, namely t_1, t_2, t_3, and t_4 where $t_1 < t_2 < t_3 < t_4$. The intervals between these times may be 1 week or 1 month, for example. T is a threshold, below which the metric values should not drop (see Section V.A). At t_1, one of the metrics was evaluated producing the value shown. At t_2, the metric was again evaluated, but this time the value was found to be much lower than at t_1, and in fact, the value dropped below T. The reason for this was that new vulnerabilities were found that had not been secured. The organization decides to secure the additional vulnerabilities. At t_3, another evaluation was carried out, and this time, the metric had improved, reaching above T. The organization finds some surplus money in its budget and decides to secure 2 other vulnerabilities. An evaluation of the metric at t_4 finds the value a little higher than at t_3, due to the 2 additional vulnerabilities secured. It is thus seen that the security level of a computer system changes over time, in accordance with the system's number of secured and unsecured vulnerabilities.

E. *Strengths, Weaknesses, and Limitations*

Some strengths of the metrics are: a) conceptually straightforward, and easily explainable to management, and b) flexible and powerful, i.e., they have many application areas, as described in Section V.

Some weaknesses are: a) threat modeling to determine the vulnerabilities is time consuming and subjective, and b) the SL will involve more factors than vulnerabilities and secured vulnerabilities. Moreover, as mentioned above, it is next to impossible to find all the p(t), so the SL determined by the metrics can never be the true SL. For weakness a), it may be possible to automate or semi-automate the threat modeling. Related works [18] and [28] are good starting points for further research. For weakness b), it may be argued that the metrics as presented are sufficient for their envisaged application when other sources of error are considered (e.g., it is difficult to tell where an attacker will strike or how he will strike), and that adding more factors would only make the metrics unnecessarily more cumbersome and time consuming to evaluate with little additional benefit. It is next to impossible to determine the true SL anyway.

Some mathematical limitations of the metrics follow. First of all, the metrics are only estimates of the security level, not the security level itself (and can never be the true SL due to unknowable p(t) as mentioned above). This was indicated in assigning the probabilities as approximate in expressions (2) and (6) above. Second, as noted in Section III.A, it makes no difference to the values of the metrics whether one unsecured vulnerability is more likely to be exploited than another. This means that the metrics are insensitive to one exploited vulnerability causing more damage than others, and may be due to the fact that the metrics are estimating the total security of the computer system, and therefore the total number of exploitable vulnerabilities is what's important, not whether a particularly damaging vulnerability is exploited. Third, we applied the additive rule for the union of probabilities above, requiring that 2 or more exploits do not occur simultaneously. This condition holds in general but if it is violated, the metrics will be inaccurate. This may not be very significant, since they are only estimates. An additional limitation may be that a secured vulnerability may not in reality be secured because the attacker has a secret way of defeating the securing measure. However, this additional limitation is true of other security methods as well.

IV. Verification of Soundness

This section examines the soundness (as defined below) of the proposed metrics using a procedure in this author's previously published paper [9]. In that paper, this author pointed out some flaws that can unintentionally be included in the definition of security metrics, leading to invalid conclusions. The flaws can be found in a number of existing security metrics that were presented in [9]. This author then proposed a procedure that can be used to design "good" or sound security metrics that would be free of the flaws. As it turns out, the procedure can also be used to check existing security metrics to verify that they are sound.

Consider the metric *number of viruses detected and eliminated at a firewall*. The purpose of this metric is to assess the effectiveness of a firewall at filtering out viruses, which impacts the organization's level of security. Unfortunately, this metric says nothing about the viruses that were not detected and got through. If 50 viruses were detected and eliminated but 100 got through, basing the firewall's effectiveness solely on the 50 viruses that were detected and not on the 100 that got through would falsely inflate the firewall's effectiveness and the level of security. Thus, this metric fails its purpose. Another often-used security metric is *time spent on a security-related task*, such as software patching or security incident investigation. The purpose of this metric is to gauge the level of security, assuming that more time spent means higher security. This metric may be useful for project management, to make sure that there is sufficient time to complete the project, but it is practically useless as an indicator of security. The assumption is wrong: more time spent does not necessarily mean better security. For example, the extra time may have been due to inefficient procedures or work processes. Thus, this metric also fails its purpose. To avoid problematic metrics such as the foregoing, this author proposed the following procedure [9] for designing good or sound security metrics.

A. *Steps for Designing Sound Security Metrics (SDSSM)*

1. **Definition:** Define the quantity to be measured, i.e. the candidate metric. Check that this quantity is meaningful, objective, and unbiased as a measure of the component or components of the security level of "something", where that "something" could be the organization, the organization's computer system, or even a software product. Check also that this quantity can be obtained with undue hardship or costs. If the quantity passes all these checks, proceed to Step 2. Otherwise, repeat this step to obtain a new quantity. Note that the quantity can only measure a *component* or *components* of the security level since the actual security level has many components, such as the number of unsecured vulnerabilities, security flaws in software, disgruntled employees, and so on. An example quantity is *number of software security patches issued in a month*, which is a component of the security level of the software.
2. **Sufficiency:** Verify that the quantity is a sufficient measure of the component or components of the security level (as in necessary and sufficient conditions for

something to be true, see [10]). It is enough to verify sufficiency since there are usually many ways to measure a component, so necessity will not apply in most cases. Verify sufficiency by asking and answering the questions in Table I. For the quantity to be sufficient, the answers to questions 1, 2, and 3 must be "yes", "yes", and "no" respectively. If the quantity is found to be sufficient, proceed to Step 3. Otherwise, repeat from Step 1 to obtain a new quantity. For example, the quantity *time spent on a security-related task* is not a sufficient estimator since spending more time does not mean that the security level will be consistently higher (or lower), as discussed above. Thus, the answer is "no" to question 1. Since this answer must be "yes" for sufficiency, this quantity is not sufficient.

TABLE I. QUESTIONS FOR DETERMINING SUFFICIENCY

No.	Question
1	If the quantity goes up, do you believe that the security level consistently goes up (or down)?
2	Does the quantity have a direct impact on the security level?
3	Are there any aspects missing from the definition of the quantity that are needed for it be effective as a measure of the component or components of the security level?

3. **Divisibility:** Verify if the quantity is divisible into other constituent quantities, or is expressible mathematically in terms of other constituent quantities. If not, proceed to step 4. Otherwise, formulate a mathematical expression that equates the quantity to the constituent quantities, and proceed to Step 4. For example, the quantity *number of software security patches issued in a month* is not further divisible, whereas the quantity *outstanding vulnerabilities after threat analysis each month* may be divided into and equated to *the number of non-secured vulnerabilities from last month* plus *the number of new vulnerabilities found during threat analysis*.
4. **Progression:** Verify that the quantity has the "progression property", that when evaluated over a sufficiently large time period, from past to future, the quantity progresses to an acceptable target level that corresponds to an acceptable or maximal security level. If the quantity has this property, proceed to Step 5. Otherwise, repeat from Step 1 to obtain a new quantity. For example, in the case of *number of software security patches issued in a month,* suppose that this metric is evaluated at the first of the month for the last month. Suppose that the target level for the quantity is zero. Thus, over a sufficiently large number of months in which patches are issued, there are corresponding increases in the security level of the software toward some level. The security level of the software increases with each patch issued until at some point, there is consistently no new patch issued (target zero reached). At this point, the security level of the software is maximal (but not necessarily maximized since there may still be undiscovered security bugs). The quantity has progressed to its target level with corresponding maximal security.
5. **Reproducibility:** Verify that the quantity is reproducible by third-party verifiers. This means that the latter may evaluate the quantity or arrive at its value using the same inputs or procedure and obtain the same result. If the quantity is reproducible, stop. The quantity is now considered a sound security metric. Otherwise, repeat from Step 1 to obtain a new quantity. For example, if the quantity is *number of software security patches issued in a month*, a third-party verifier would add up the software security patches issued for a particular month, and find the same number as the organization that is using the metric. If the quantity is *outstanding vulnerabilities after threat analysis each month*, which we know is equated to *the number of non-secured vulnerabilities from last month* plus *the number of new vulnerabilities found during threat analysis*, the third-party verifier would do the latter addition and verify that the total is the same as obtained by the organization using the metric.

Procedure SDSSM can be used not only to design a sound security metric but also to verify if an existing security metric is sound. This verification is carried out by checking if the metric satisfies each of the steps in SDSSM except for STEP 3, which is not a condition to be checked. STEP 3 is only used when designing a security metric, in order to allow the metric to take on a clearer form. This verification of soundness is captured in the following definition.

DEFINITION 5: A security metric is *sound* if it satisfies every step in SDSSM, excluding STEP 3.

We now apply definition 5 to verify that the metrics proposed in Section III are sound. These metrics are:

$$\begin{aligned}\mathbf{STRM(t)} &= \mathbf{q(t)/(p(t)+q(t))} && \textbf{if } \mathbf{p(t)+q(t) > 0}\\ &= \mathbf{1} && \textbf{if } \mathbf{p(t)+q(t) = 0}\end{aligned}$$

$$\begin{aligned}\mathbf{SPRM(t)} &= \mathbf{\Sigma_k r_k(t)/(p(t)+q(t))} && \textbf{if } \mathbf{p(t) \geq 0,\ q(t) > 0}\\ &= \mathbf{1} && \textbf{if } \mathbf{p(t)+q(t) = 0}\\ &= \mathbf{0} && \textbf{if } \mathbf{p(t)>0,\ q(t)=0}\end{aligned}$$

It suffices to check that these metrics satisfy the conditions in each step of SDSSM, as follows.

STEP 1: Definition. The security of the computer system is directly related to the number of secured vulnerabilities in the system: the higher this number, the higher the security, and the lower this number, the lower the security. Consequently, since both metrics express the security level in terms of the proportion of secured vulnerabilities to total vulnerabilities, both metrics are clearly meaningful for assessing the security level (note that the numerator in SPRM(t) is really the number of secured vulnerabilities as a fractional or real number). The metrics are objective since secured vulnerabilities relate directly to the security of the system. They are unbiased since their values, based on secured and unsecured vulnerabilities, cannot be overstated or understated. Finally, one can evaluate these metrics without undue hardship or cost by doing a vulnerability or threat analysis, deciding which vulnerabilities to secure, and using the reliabilities of the securing measures where available. Thus, these metrics are considered to have passed Step 1 and we proceed to Step 2.

STEP 2: Sufficiency. We answer the questions in Table I. The first question asks if the security would consistently go up (or down) if the quantity (metric) goes up. Clearly if the value of STRM(t) goes up, the number of secured vulnerabilities must consistently go up since the denominator is a constant. In other words, the security consistently goes up. The same can be said of SPRM(t), since its numerator is the number of secured vulnerabilities as a real number. So, the answer to the first question is "yes" for both metrics. The second question asks if the quantity has a direct impact on the security level. The answer is again "yes" for both metrics, since the higher their values, the higher the security level, and the lower their values, the lower the security level. Finally, the third question asks if the quantity is missing any components or aspects that are needed for it to be effective. The answer here is "no" for both metrics, since they are ready to be used "as is" for effectively assessing the security level. The answers to the three questions conform to the answers required for sufficiency. We declare the metrics sufficient and proceed to Step 4, since STEP 3 is not needed for verifying soundness.

STEP 4: Progression. Suppose that vulnerabilities are determined (through a threat analysis) and one of the metrics (STRM(t) if no reliability values are available, SPRM(t) otherwise) is re-calculated at regular time intervals, e.g., monthly. Suppose also that Company A's management has agreed on a goal of 95% for the metric, at which level the computer system is considered "safe", i.e. management is willing to live with the risks arising from the remaining non-secured vulnerabilities. With this goal in mind, management will want to secure vulnerabilities at each opportunity until the metric attains 95%. This doesn't mean that the metric will increase monotonically, since it is possible that a particular threat analysis identifies so many new vulnerabilities that the metric is actually lower than when it was last calculated. However, the metric will eventually reach 95%, given that management wants to secure new vulnerabilities until this goal is reached, which is all we mean by having the progression property. Since this analysis applies to both metrics, we can consider them as having passed Step 4 and proceed to Step 5.

STEP 5: Reproducibility. Given the expression for STRM(t), anyone will calculate the same value for it given the same values for p(t) and q(t). Similarly, given the expression for SPRM(t), anyone will calculate the same value for it given the same values for the reliabilities, p(t), and q(t). Thus, the metrics are reproducible.

Thus, according to Definition 5, the metrics STRM(t) and SPRM(t) are sound.

To show that the application of SDSSM can find that a metric is not sound, consider its application to the flawed metric mentioned above, namely the metric *number of viruses detected and eliminated at a firewall.* Applying SDSSM to this metric leads to it failing STEP 1 Definition, since it is biased towards overstating the firewall's effectiveness. Thus, according to Definition 5, this metric is not sound. Note that the metric *time spent on a security-related task* would also be found by SDSSM as not sound since it failed STEP 2 Sufficiency, as indicated in the description of SDSSM above.

V. Application Areas

In this section, we present some applications for the metrics. In Section V.A, we discuss how they can be used for continuous active defence of a computer system. In Section V.B, we present other application areas, such as critical infrastructure and defence.

A. *Continuous Active Defence*

Attackers do not attack once, and finding that you are well protected, go away. Rather, they continuously probe your defences in order to find new vulnerabilities to exploit. It is thus necessary to continuously evaluate the computer system's vulnerabilities using threat modeling, and add additional security by securing new vulnerabilities when necessary. We call this "Continuous Active Defence" or CAD. How do we know when it is necessary to add more security? This is where the metrics can be applied. Continuous Active Defence involves the following steps:

1. Decide on a threshold for SL(t) below which the values of the metrics should not drop.
2. Decide on the frequency with which to perform threat modeling, e.g., every week, every month, exceptions.
3. Begin Continuous Active Defence by carrying out the threat modeling at the frequency decided above. After each threat modeling exercise, calculate either STRM(t) (if reliability data is not available) or SPRM(t) (if reliability data is available). If the value of the metric falls below T (see Figure 2), secure additional vulnerabilities until the value is above T.
4. If there has been a change to the system, such as new equipment or new software, do an immediate threat analysis, calculate one of the metrics, and add security if necessary based on T. Then, proceed with the frequency for threat modeling decided above.

The value of T and the frequency of threat modeling can be determined by the same threat analysis team mentioned above. The values would depend on the following:

- The potential value of the sensitive data – the more valuable the data is to a thief, a malicious entity, or a competitor, the higher the threshold and frequency should be.
- The damages to the organization that would result, if the sensitive data were compromised – of course, the higher the damages, the higher the threshold and frequency.
- The current and likely future attack climate – consider the volume of attacks and the nature of the victims, say over the last 6 months; if the organization's sector or industry has sustained a large number of recent attacks, then the threshold and frequency need to be higher.
- Consider also potential attacks by nation states as a result of the political climate; attacks by individual hacktivist groups such as Anonymous or WikiLeaks may also warrant attention.

In general, a computer system should be as secure as possible. Therefore, T above 80% and a frequency of weekly would not be uncommon. However, whatever the threshold and frequency, the organization must find them acceptable after considering the above factors. The financial budget available for securing vulnerabilities also plays an important role here, since higher thresholds call for securing more vulnerabilities, which means more financial resources will be needed.

B. *Other CAD Application Areas*

CAD may also be applied to a specific type of vulnerabilities. An example of this application is dealing with inside attacks. If the organization is particularly susceptible to inside attacks, it can decide to apply CAD to vulnerabilities that can be exploited for inside attacks. In this case, some of the vulnerabilities may be weaknesses of the organization itself, e.g., ineffective screening of job applicants, and the securing measures may not be technological, e.g., having an ombudsman for employee concerns. A list of questions that can be used to identify vulnerabilities to inside attack is given in [8].

CAD may be applied to a specific subset of vulnerabilities that the organization deems are crucial to its mission. For example, a cloud service provider would deem the protection of clients' data crucial to its mission. It can choose to apply CAD to vulnerabilities that are specific to its data storage capabilities, and also apply CAD to its computer system as a whole.

CAD may also be applied to code level vulnerabilities. In this case, the frequency of application will depend on how often the code is changed, due to patching and the addition or deletion of functionality. The threat modeling would have to be tailored to code and would be more of a code inspection exercise.

Finally, CAD may be applied to protect critical infrastructure and defence systems. The power grid is an example of critical infrastructure. The development of the metrics only considers vulnerabilities and reliabilities, which are also found in critical infrastructure and defence systems. However, the threat analyses would involve different types of threats, and the securing measures, would of course, need to be appropriate for the vulnerability. For example, the vulnerability of transformer sabotage in a power grid may need to be secured by the use of intrusion alarms. As another example, the vulnerability of a retaliatory missile site being preemptively destroyed may need to be secured by putting the missile on a mobile platform. The application of CAD to protect these areas is a subject of future research.

C. *Where CAD May and May Not Be Applied*

Fundamentally, CAD may be applied to organizations and systems that have the following elements:

a) Possess "something" that attackers want
b) Vulnerabilities that change over time and that attackers can attack to access the "something"
c) Measures (or controls) that can be used to secure the vulnerabilities from attack

An examination of the above CAD application areas will find these elements present in each area. Organizations or systems that are missing any of these elements are therefore not suitable for the application of CAD. An example of such a "system" may be an expensive bicycle. In this case, it is the bicycle itself that thieves (attackers) want. Its vulnerability is that it can be stolen if the bicycle is not suitably secured. The measure that can be used to secure the bicycle is a strong lock. However, the bicycle's vulnerability to being stolen is not changing over time. This vulnerability will be the same always, even if the bicycle becomes less attractive to thieves over time. This bicycle is not a suitable system for the application of CAD.

VI. RELATED WORK

Related work found in the literature includes attack surface metrics, risk and vulnerabilities assessment, vulnerabilities classification, threat analysis, an "other" category, and this author's previous work. We discuss each of these categories in turn, starting with attack surface metrics.

A system's attack surface is related to a SL; it is proportional to the inverse of a SL since the lower the attack surface, the higher the SL. Manadhata and Wing [11] formalize the concept of a system's attack surface and propose an attack surface metric for systematically measuring the attack surface. They claim that their metric does not depend on the software system's implementation language and can be used on systems of all sizes. They further provide demonstrations of the metric and have conducted empirical studies to validate it. Stuckman and Purtilo [12] present a framework for formalizing code-level attack surface metrics and describe activities that can be carried out during application deployment to reduce the application's attack surface. They also describe a tool for determining the attack surface of a web application, together with a method for evaluating an attack surface metric over a number of known vulnerabilities. Munaiah and Meneely [13] propose function and file level attack surface metrics that allow fine-grained risk assessment. They claim that their metrics are flexible in terms of granularity, perform better than comparable metrics in the literature, and are tunable to specific products to better assess risk.

In terms of risk and vulnerabilities assessment, Islam et al. [14] present a risk assessment framework that starts with a threat analysis followed by a risk assessment to estimate the threat level and the impact level. This leads to an estimate of a security level for formulating high-level security requirements. The security level is qualitative, such as "low", "medium", and "high". Vanciu et al. [15] compare an architectural-level approach with a code-level approach in terms of the effectiveness of finding security vulnerabilities. Wang et al. [16] discuss their work on temporal metrics for software vulnerabilities based on the Common Vulnerability Scoring System (CVSS) 2.0. They use a mathematical model to calculate the severity and risk of a vulnerability, which is time dependent as in this work. Gawron et al. [17] investigate the detection of vulnerabilities in computer systems and computer networks. They use a logical representation of preconditions and post conditions of vulnerabilities, with the aim of providing security advisories and enhanced diagnostics for the system. Wu and Wang [18] present a dashboard for assessing enterprise level vulnerabilities that incorporates a multi-layer tree-based model to describe the vulnerability topology. Vulnerability information is gathered from enterprise resources for display automatically. Farnan and Nurse [19] describe a structured approach to assessing low-level infrastructure vulnerability in networks. The approach emphasizes a controls-based evaluation rather than a vulnerability-based evaluation. Instead of looking for vulnerabilities in infrastructure, they assume that the network is insecure, and determine its vulnerability based on the controls that have or have not been implemented. Neuhaus et al. [20] present an investigation into predicting vulnerable software components. Using a tool that mines existing vulnerability databases and version archives, mapping past vulnerabilities to current software components, they were able to come up with a predictor that correctly identifies about half of all vulnerable components, with two thirds of the predictions being correct. Roumani et al. [21] consider the modeling of vulnerabilities using time series. According to these researchers, time series models provide a good fit to vulnerability datasets and can be used for vulnerability prediction. They also suggest that the level of the time series is the best estimator for prediction. Li et al. [22] present VulPecker, a tool for automatically detecting whether source code contains a particular vulnerability. Pang et al. [23] propose a technique based on a deep neural network to predict vulnerable software components. They claim that their technique can predict vulnerable Java classes in Android applications with high accuracy. Anand et al. [24] propose a model for classifying security patterns according to the type of vulnerability they address, claiming that their model helps software developers to select an appropriate security pattern once they know the type of vulnerability they would like to remove. The authors also claim that their classification scheme identifies missing security patterns, when no patterns can be found for particular vulnerabilities. Salfer and Eckert [25] consider the attack surface and vulnerability assessment of automotive electronic control units (ECUs). They propose a method and metric for assessing the attack surface and predicting the effort for a code injection exploit using ECU development data. They also provide an application of their method and metric to a graph-based security assessment.

With regard to vulnerabilities classification, Spanos et al. [26] look at ways to improve CVSS. They propose a new vulnerability scoring system called the Weighted Impact Vulnerability Scoring System (WIVSS) that incorporates the different impact of vulnerability characteristics. In addition, the MITRE Corporation [27] maintains the Common Vulnerability and Exposures (CVE) list of vulnerabilities and exposures, standardized to facilitate information sharing.

In terms of threat analysis, Schaad and Borozdin [28] present an approach for automated threat analysis of software architecture diagrams. Their work gives an example of automated threat analysis. Sokolowski and Banks [29] describe the implementation of an agent-based simulation model designed to capture insider threat behavior, given a set of assumptions governing agent behavior that pre-disposes an agent to becoming a threat. Sanzgiri and Dasgupta [30] present a taxonomy and classification of insider threat detection techniques based on strategies used for detection. Manzoor et al. [31] claim that contemporary cloud threat analysis approaches fail to include variants of identified vulnerabilities in their analyses. They target achieving a holistic cloud threat analysis procedure by designing a multi-layer cloud model, employing Petri Nets to comprehensively profile the operational behavior of the services in cloud operations. They use this model to identify threats within and across different operational layers. They further claim that their approach also looks at the variants of potential vulnerabilities to infer the cloud attack surface. Valani [32] looks at Secure DevOps threat modeling and concludes that maintaining speed to support business needs is difficult due to the fact that the threat modeling is too slow. He proposes the use of a lightweight threat modeling approach that uses a correlation matrix created from common lists and application abstractions, that is quicker and can be applied where detailed threat modeling is unnecessary.

The following publications fall into the other category. Kotenko and Doynikova [33] investigate the selection of countermeasures for ongoing network attacks. They suggest a selection technique based on the countermeasure model in open standards. The technique incorporates a level of countermeasure effectiveness that is related to the reliability of measures securing vulnerabilities, used in the SPRM(t) metric proposed in this work. Ganin et al. [34] present a review of probabilistic and risk-based decision-making techniques applied to cyber systems. They propose a decision-analysis-based approach that quantifies threat, vulnerability, and consequences through a set of criteria designed to assess the overall utility of cybersecurity management alternatives. Pendleton et al. [35] provide a systematic survey of systems security metrics. Based on this survey, they propose that an overall system security metric can be represented by the following dimensions of metrics: vulnerabilities, defenses, attacks, and situations. The situation dimension is focused on the current security state of a given system at a particular point in time, in order to account for dynamics related to system security states, including the level of vulnerabilities, attacks, and system defenses.

This author's directly related work includes [36], [8], and [1] where [8] is an expanded version of [36]. Yee [1] improves on [36] and [8] by a) adding time dependency, together with the notion that an organization's security level needs to be continuously evaluated, b) adding a new metric incorporating the reliability of the securing measures, and c) adding a description of new application areas. This work extends Yee [1] by adding the material mentioned at the start of Section I.

VII. Conclusion and Future Work

Since attackers continuously probe for new vulnerabilities to exploit, an organization cannot afford to assess its computer system's vulnerabilities once, secure some of the vulnerabilities, and then do nothing further.

Rather, the organization needs to assess and secure its vulnerabilities on a continuous basis, i.e., perform CAD. This work has proposed two conceptually clear SL metrics, verified as sound, that can be used to evaluate a computer system's security level at any point in time for CAD. One metric assumes that the measures securing vulnerabilities are totally reliable; the other considers the measures to be only partially reliable. CAD may be applied to specific types of vulnerabilities (e.g., vulnerabilities to insider attack), groupings of vulnerabilities that require special attention, specific application areas such as critical infrastructure and defence, and even at the code level. CAD may not be applied to areas that are missing any of the elements listed in Section V.C.

There are many security metrics in the literature, as seen in Section VI. The metrics in this work have the advantages of being easy to understand, and easy to calculate, which may be needed to convince management to provide the necessary resources required for CAD.

Future work includes formulations of other security metrics, the application of security metrics to critical infrastructure and defence, improving the methods for threat modeling, and exploring how this work may complement work in the literature and in the standardization community.

REFERENCES

[1] G. Yee, "Metrics for continuous active defence," Proc. Twelfth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2018), pp. 92-98, 2018.

[2] D. Rafter, "2019 Data Breaches: 4 Billion Records Breached So Far," Norton, retrieved December, 2019 from: https://us.norton.com/internetsecurity-emerging-threats-2019-data-breaches.html

[3] Identity Force, "2018 Data Breaches – The Worst of Last Year," retrieved November, 2019 from: https://www.identityforce.com/blog/2018-data-breaches

[4] Dark Reading, "2017 Smashed world's records for most data breaches, exposed information," retrieved November, 2019 from: https://www.darkreading.com/attacks-breaches/2017-smashed-worlds-records-for-most-data-breaches-exposed-information/d/d-id/1330987?elq_mid=83109&elq_cid=1734282&_mc=NL_DR_EDT_DR_weekly_20180208&cid=NL_DR_EDT_DR_weekly_20180208&elqTrackId=700ff20d23ce4d3f984a1cfd31cb11f6&elq=5c10e9117ca04ba0ad984c11a7dfa14b&elqaid=83109&elqat=1&elqCampaignId=29666

[5] G. Yee, "Visualization and prioritization of privacy risks in software systems," International Journal on Advances in Security, issn 1942-2636, vol. 10, no. 1&2, pp. 14-25, 2017.

[6] ITEM Software Inc.,"Reliability prediction basics", retrieved November, 2019 from: http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf

[7] C. Salter, O. Saydjari, B. Schneier, and J. Wallner, "Towards a secure system engineering methodology," Proc. New Security Paradigms Workshop, pp. 2-10, 1998.

[8] G. Yee, "Optimal security protection for sensitive data," International Journal on Advances in Security, vol. 11, no. 1&2, pp. 80-90, 2018.

[9] G. Yee, "Designing good security metrics," Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, pp. 580-585, July 15-19, 2019.

[10] N. Swartz, "The concepts of necessary conditions and sufficient conditions," Department of Philosophy, Simon Fraser University, 1997, retrieved November, 2019 from: https://www.sfu.ca/~swartz/conditions1.htm

[11] P. K. Manadhata and J. M. Wing, "An attack surface metric," IEEE Transactions on Software Engineering, vol. 37, no. 3, pp. 371-386, May/June, 2011.

[12] J. Stuckman and J. Purtilo, "Comparing and applying attack surface metrics," Proceedings of the 4th International Workshop on Security Measurements and Metrics (MetriSec '12), pp. 3-6, Sept. 2012.

[13] N. Munaiah and A. Meneely, "Beyond the attack surface," Proceedings of the 2016 ACM Workshop on Software Protection (SPRO '16), pp. 3-14, October 2016.

[14] M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," Proc. 2nd ACM International Workshop on Cyber-Physical System Security (CPSS '16), pp. 3-14, 2016.

[15] R. Vanciu, E. Khalaj, and M. Abi-Antoun, "Comparative evaluation of architectural and code-level approaches for finding security vulnerabilities," Proceedings of the 2014 ACM Workshop on Security Information Workers (SIW '14), pp. 27-34, Nov. 2014.

[16] J. A. Wang, F. Zhang, and M. Xia, "Temporal metrics for software vulnerabilities," retrieved: November, 2019. http://www.cs.wayne.edu/fengwei/paper/wang-csiirw08.pdf

[17] M. Gawron, A. Amirkhanyan, F. Cheng, and C. Meinel, "Automatic vulnerability detection for weakness visualization and advisory creation," Proc. 8th International Conference on Security of Information and Networks (SIN '15), pp. 229-236, 2015.

[18] B. Wu and A. Wang, "A multi-layer tree model for enterprise vulnerability management," Proceedings of the 2011 Conference on Information Technology Education (SIGITE '11), pp. 257-262, October 2011.

[19] O. Farnan and J. Nurse, "Exploring a controls-based assessment of infrastructure vulnerability," Proc. International Conference on Risks and Security of Internet and Systems (CRiSIS 2015), pp. 144-159, 2015.

[20] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," Proc. 14th ACM Conference on Computer and Communications Security (CCS '07), pp. 529-540, 2007.

[21] Y. Roumani, J. Nwankpa, and Y. Roumani, "Time series modeling of vulnerabilities," Computers and Security, Vol. 51 Issue C, pp. 32-40, June 2015.

[22] Z. Li, D. Zou, S. Xu, H. Jin, H. Qi, and J. Hu, "VulPecker: an automated vulnerability detection system based on code similarity analysis," Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC'16), pp. 201-213, Dec. 2016.

[23] Y. Pang, X. Xue, and H. Wang, "Predicting vulnerable software components through deep neural network," Proceedings of the 2017 International Conference on Deep Learning Technologies (ICDLT'17), pp. 6-10, June 2017.

[24] P. Anand, J. Ryoo, and R. Kazman, "Vulnerability-based security pattern categorization in search of missing patterns," Proceedings of the 2014 Ninth International Conference on Availability, Reliability and Security (ARES), pp. 476-483, Sept. 2014.

[25] M. Salfer and C. Eckert, "Attack surface and vulnerability assessment of automotive electronic control units," Proceedings of the 12th International Conference on Security and Cryptography (SECRYPT 2015), pp. 317-326, 2015.

[26] G. Spanos, A. Sioziou, and L. Angelis, "WIVSS: A new methodology for scoring information system vulnerabilities," Proc. 17th Panhellenic Conference on Informatics, pp. 83-90,

2013.

[27] MITRE, “Common vulnerabilities and exposures”, retrieved November, 2019 from: https://cve.mitre.org/

[28] A. Schaad and M. Borozdin, “TAM2: Automated threat analysis,” Proc. 27th Annual ACM Symposium on Applied Computing (SAC ’12), pp. 1103-1108, 2012.

[29] J. Sokolowski and C. Banks, “An agent-based approach to modeling insider threat,” Proc. Symposium on Agent-Directed Simulation (ADS '15), pp. 36-41, 2015.

[30] A. Sanzgiri and D. Dasgupta, “Classification of insider threat detection techniques,” Proc. 11th Annual Cyber and Information Security Research Conference (CISRC ’16), article no. 25, pp. 1-4, 2016.

[31] S. Manzoor, H. Zhang, and N. Suri, “Threat modeling and analysis for the cloud ecosystem,” Proceedings of the 2018 IEEE International Conference on Cloud Engineering, pp. 278-281, 2018.

[32] A. Valani, “Rethinking Secure DevOps threat modeling: the need for a dual velocity approach,” Proceedings of the 2018 IEEE Secure Development Conference (SecDev), pp. 136, 2018.

[33] I. Kotenko and E. Doynikova, “Dynamical calculation of security metrics for countermeasure selection in computer networks,” Proc. 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 558-565, 2016.

[34] A. Ganin, P. Quach, M. Panwar, Z. A. Collier, J. M. Keisler, D. Marchese, and I. Linkov, “Multicriteria decision framework for cybersecurity risk assessment and management,” Risk Analysis, pp. 1-17, 2017.

[35] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, “A Survey on Systems Security Metrics,” ACM Computing Surveys (CSUR), Vol. 49, Issue 4, Article No. 62, pp. 1-35, February 2017.

[36] G. Yee, “Assessing security protection for sensitive data,” Proc. Eleventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2017), pp. 111-116, 2017.