Aggregating CVSS Base Scores for Semantics-Rich Network Security Metrics

Pengsu Cheng*, Lingyu Wang*, Sushil Jajodia† and Anoop Singhal‡
*Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 1M8, Canada
Email: {pen_che,wang}@ciise.concordia.ca
†Center for Secure Information Systems, George Mason University, Fairfax, VA 22030-4444, USA
Email: jajodia@gmu.edu
‡Computer Security Division, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
Email: anoop.singhal@nist.gov

***Abstract*—A network security metric is desirable in evaluating the effectiveness of security solutions in distributed systems. Aggregating CVSS scores of individual vulnerabilities provides a practical approach to network security metric. However, existing approaches to aggregating CVSS scores usually cause useful semantics of individual scores to be lost in the aggregated result. In this paper, we address this issue through two novel approaches. First, instead of taking each base score as an input, our approach drills down to the underlying base metric level where dependency relationships have well-defined semantics. Second, our approach interprets and aggregates the base metrics from three different aspects in order to preserve corresponding semantics of the individual scores. Finally, we confirm the advantages of our approaches through simulation.**

I. Introduction

Today's critical infrastructures and enterprises are increasingly dependant on the reliable functioning of distributed systems. In securing such systems, a network security metric is desirable since *you cannot improve what you cannot measure*. By applying a security metric immediately before, and after, deploying security solutions, we can judge those solutions' relative effectiveness in a direct and precise manner. Such a capability will make securing networks a science, rather than an art.

The Common Vulnerability Scoring System (CVSS) is a widely adopted standard [12], which allows security analysts and vendors to assign numerical scores to vulnerabilities based on their relative severity. CVSS scores of known vulnerabilities are already available through public vulnerability databases (e.g., the NVD [13]). CVSS thus provides a practical foundation for developing network security metrics. On the other hand, CVSS is mainly intended for ranking individual vulnerabilities. It does not directly provide a way for aggregating individual scores into an overall metric of network security. Naive ways for aggregating scores (e.g., taking the average or maximum value) usually lead to misleading results, whereas existing attack graph-based approaches can achieve improved results [6], [5], [18].

In this paper, we first observe that most existing approaches to aggregating CVSS scores may cause useful semantics of individual scores to be lost in two ways. First, vulnerabilities' dependency relationship is currently either ignored or handled in an arbitrary way, which brings doubts to the metric results and prevents their adoption. Instead of taking the base score as a black box input, we break it down to the underlying base metrics in which dependency relationships have well-defined semantics. Second, only the attack probability is currently considered for aggregating CVSS scores, which may limit the scope of application and lead to misleading results since different aspects will demand different algebra for aggregating the scores. To address this issue, we interpret and aggregate CVSS scores from three aspects, namely, probability, effort, and skill. Finally, we confirm the advantages of our approach through simulation.

The contribution of this paper is three-fold. First, the novel approach of base metric-level aggregation can preserve more semantics and consequently produce more meaningful metric results. Second, interpreting and aggregating CVSS scores from different aspects allows different semantics to be extracted from CVSS scores, and consequently may broaden the scope of application for the CVSS standard. Third, to the best of our knowledge, the simulation presented in this paper is among the first efforts on numerically evaluating security metrics.

The rest of this paper is organized as follows. Section II reviews background information. Section III presents the base metric-level aggregation. Section IV addresses three different aspects. Section V presents simulation results. Finally, Section VI reviews related work and Section VII concludes the paper.

II. Preliminaries

We first briefly review the CVSS standard to make our paper more self-contained. We then demonstrate limitations of existing approaches through an example.

A. The CVSS Standard

In CVSS, each vulnerability is assigned a *base score (BS)* ranging from 0 to 10, based on two groups of totally six *base metrics* [12]. These base metrics will stay constant over time, and across different user environments. Optionally, the base score can be adjusted with temporal and environmental scores to reflect time or application-specific factors (the temporal and environmental scores are generally not available in vulnerability databases, and are not considered in this paper). Specifically,

- The $Exploitability$ metric group measures the relative difficulty to exploit a vulnerability:
  - $AccessVector$ measures the distance from which the vulnerability can be accessed. Possible values include $Local$ (e.g., physical access required), $AdjacentNetwork$ (e.g., accessible from local subnets), and $Network$ (e.g., accessible from the Internet).
  - $AccessComplexity$ measures the complexity to exploit the vulnerability once an attacker has the required access. Possible values include $High$ (e.g., admin/root privilege required), $Medium$(e.g., user privilege required), Low (e.g., exploitable with default account).
  - $Authentication$ measures the amount of required authentication effort. Possible values include $Multiple$ (e.g., authentications required at both OS and applications), $Single$ (e.g., authentication required at OS only), and $None$ (e.g., no authentication required).
- The $Impact$ group measures the potential consequences of exploiting a vulnerability. For each metric in this group, the possible values are $None$ (e.g., no impact to confidentiality), $Partial$ (e.g., modification of some files possible), and $Complete$ (e.g., resource rendered completely unavailable).

The six base metrics will be mapped to fixed numerical values (details omitted) and used to calculate the base score (BS) with the *base equation* (see Equation 1).

B. Limitations of Existing Approaches

Figure 1 shows a toy network with two hosts (1 and 2) on different subnets and an attacker's host 0 in the Internet. We consider two cases based on this network. In Case 1, we assume host 1 to be a UNIX server running a telnet service and host 2 a Windows XP workstation running the Universal Plug and Play (UPnP) service. In Case 2, host 1 and 2 swap their OS (and corresponding services). In both cases, the firewalls disallow any traffic except accesses to those services.

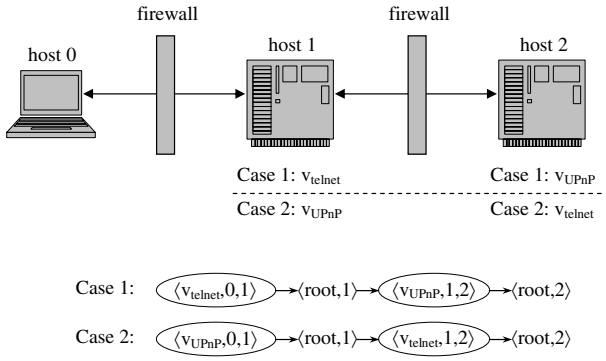


Fig. 1. An Example Network

We assume the telnet service contains the vulnerability CVE-2007-0956 [13], denoted by v_{telnet}, which allows remote attackers to bypass authentication and gain system accesses via providing special usernames to the service. Also, the UPnP service contains the vulnerability CVE-2007-1204 [13], denoted by v_{UPnP}, which is a stack overflow that allows attackers on the same subnet to execute arbitrary codes via sending specially crafted requests. Table I shows their CVSS base metrics [13]. By applying Equation 1, we can calculate the base score $BS = 7.6$ for v_{telnet} and $BS = 6.8$ for v_{UPnP}.

a) Average and Maximum: First, consider two naive ways for aggregating the CVSS scores, taking the average value (7.2 in both Case 1 and 2) and maximum value (7.6 in both cases), respectively. Since the average and maximum values are both defined over a set, they do not depend on where vulnerabilities are located in a network, or how they are related to each other. For example, if we assume the UNIX server in Figure 1 to be the only important asset, then intuitively the overall security is quite different between Case 1 (an attacker can directly attack the UNIX server on host 1) and Case 2 (he/she must first compromise the Windows workstation on host 1 before attacking host 2). Nonetheless, by taking the average or maximum value, we cannot distinguish between the two cases.

b) Attack Graph-Based Approach [18]: The above naive approaches lead to misleading results because they ignore causal relationships between vulnerabilities. Such causal relationships can be modeled in attack graphs, as illustrated in the lower portion of Figure 1. Each triple $\langle v, h_1, h_2 \rangle$ inside an oval represents an exploit of vulnerability v on host h_2 from host h_1; each pair $\langle c, h \rangle$ represents a security-related condition c on host h.

$$
\begin{aligned}
BS &= round\_to\_1\_decimal((0.6 * Impact + 0.4 * Exploitability - 1.5) * f(Impact)) \\
Impact &= 10.41 * (1 - (1 - ConfImpact) * (1 - IntegImpact) * (1 - AvailImpact)) \\
Exploitability &= 20 * AccessVector * AccessComplexity * Authentication \\
f(Impact) &= 0 \ if \ \ Impact = 0, \ 1.176 \ \ otherwise
\end{aligned} \tag{1}
$$

Metric Group	Metric	Metric Value of v_{telnet}	Metric Value of v_{UPnP}
Exploitability	Access Vector	Network (1.00)	Adjacent Network (0.646)
	Access Complexity	High (0.35)	High (0.35)
	Authentication	None (0.704)	None (0.704)
Impact	Confidentiality	Complete (0.660)	Complete (0.660)
	Integrity	Complete (0.660)	Complete (0.660)
	Availability	Complete (0.660)	Complete (0.660)
Base Score (BS)		7.6	6.8

TABLE I
THE CVSS BASE METRICS AND SCORES OF TWO VULNERABILITIES

The attack graph-based approach [18] converts the CVSS base scores into probabilities. The probabilities are then aggregated based on following causal relationships: An exploit is reachable only if all of its preconditions are satisfied (that is, a conjunction); a condition is satisfied as long as one reachable exploit has that condition as its post-condition (that is, a disjunction).

In Case 1 of our example, we would assign $7.6/10 = 0.76$ to $\langle v_{telnet}, 0, 1\rangle$, and $6.8/10 = 0.68$ to $\langle v_{UPnP}, 1, 2\rangle$ (and 1 to both conditions). We can then calculate the new value for $\langle root, 1\rangle$ to be 0.76 and $\langle v_{UPnP}, 1, 2\rangle$ and $\langle root, 2\rangle$ to be $0.76 \times 0.68 = 0.52$. Similarly, we will obtain the same result for Case 2. At first glance, this might seem reasonable since the attacker is exploiting the same two vulnerabilities in both cases. However, upon more careful observation, this is not the case. First, we recall that the vulnerability v_{UPnP} (CVE-2007-1204) requires the attacker to be within the same subnet as the victim host. In Case 1, exploiting v_{telnet} on host 1 helps the attacker to gain accesses to local network, and hence makes it easier to exploit host 2. In contrast, in Case 2, there is no such effect due to the reversed order of exploits. Clearly, this difference between the two cases cannot be captured by the identical result 0.52 produced by this approach.

c) Bayesian Network (BN)-Based Approach [5]: Next we consider the Bayesian network-based approach [5]. The lower left-hand side of Figure 2 shows the BN corresponding to Case 2 of our example. The lower right-hand side of Figure 2 depicts the corresponding Conditional Probability Table(CPT) for each exploit in Case 2. The probability of reaching the goal state, which is assumed as exploiting both vulnerabilities in this example, can be calculated as $P(v_{telnet} = T) = \sum_{v_{UPnP} \in \{T,F\}} P(v_{telnet} = T, v_{UPnP}) = 0.52$.

The upper left-hand side of Figure 2 depicts the BN

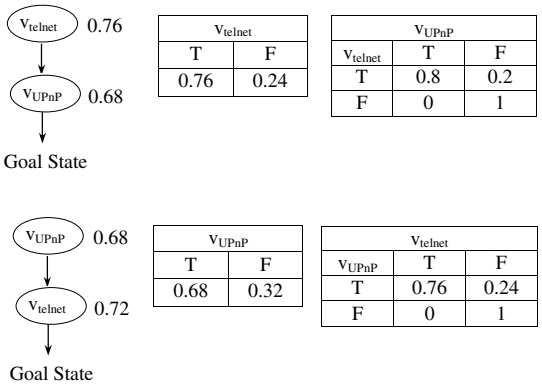


Fig. 2. Bayesian Network-Based Approach [5]

for Case 1. Since exploiting v_{telnet} on host 1 makes it easier to exploit v_{UPnP} on host 2, according to this approach, we should assign to $P(v_{UPnP} = T | v_{telnet} = T)$ a value higher than the one directly derived from the base score (that is, 0.68). If we assign, say, 0.8, then the possibility of achieving the goal state is $P(v_{UPnP} = T) = \sum_{v_{telnet} \in \{T,F\}} P(v_{UPnP} = T, v_{telnet}) = 0.61$. This result is more accurate since it reflects the dependency relationship between the two exploits. However, note that we have chosen an arbitrary value 0.8 because this approach does not provide means for determining that value, which is clearly a limitation.

III. BASE METRIC-LEVEL METRIC AGGREGATION

This section presents our approach to aggregating CVSS base metrics in order to remove the aforementioned limitations of existing approaches.

A. Overview

Our key observation is that all the existing approaches discussed in the previous section take the CVSS base scores as their inputs; the base score is regarded as a black box, and the underlying base metrics are not

involved in the process of aggregating scores. However, we notice that *the dependency relationships between vulnerabilities are usually only visible at the level of base metrics*, which prevents those approaches from properly handling such relationships.

Instead of working at the base score level, our approach drills down to the underlying base metric level. To build intuitions, we revisit the example shown in Figure 1. In that example, the dependency relationship can be easily modeled at the base metric level as follows. When an attacker successfully exploits v_{telnet} on host 1, he/she can gain accesses to the local network of host 2, which is required for exploiting v_{UPnP} on host 2. At the base metric level, this simply means the $AccessVector$ metric of v_{UPnP}, which has the value $AdjacentNetwork$, should be replaced with $Network$, since the attacker is effectively accessing v_{UPnP} remotely (using host 1 as a stepping stone).

With this adjustment to the base metric $AccessVector$, we can apply Equation 1 to recalculate a new *effective base score*, which is equal to 0.76 in this case. Clearly, similar to the BN-based approach [5], our approach also produces a result higher than the original value 0.68. However, unlike the arbitrary value chosen in [5], our result has inherited the well defined semantics from the base metrics.

The final score corresponding to Case 1 shown in Figure 1 can now be calculated as $P(v_{UPnP} = T) = \sum_{v_{telnet} \in \{T,F\}} P(v_{UPnP} = T, v_{telnet}) = 0.58$. In Table II, we summarize our discussions about the above example and compare the results produced by different approaches.

B. The Formal Approach

We are now ready to formalize our approach [1]. We assume an *attack graph* is given as a directed graph $G = \langle E \cup C, \{\langle x, y\rangle : (y \in E \wedge x \in pre(y)) \vee (x \in E \wedge y \in post(x))\}\rangle$ where E, C, $pre()$, and $post()$ denote a set of exploits (each of which is a triple $\langle v, h_s, h_d\rangle$ denoting an exploit of vulnerability v on host h_d from host h_s), a set of security-related conditions, a function that maps an exploit to the set of its pre-conditions, and a function that maps an exploit to the set of its post-conditions, respectively [1].

We call a condition *initial condition* if it is not the post-condition of any exploit. A sequence of exploits is called an *attack sequence* if for every exploit e in the sequence, all its pre-conditions are either initial conditions, or post-conditions of some exploits that appear before e in that sequence. We say an exploit e' is an *ancestor* of another exploit e, if e' appears before e in at least one minimal attack sequence (that is, an attack sequence of which no subsequence is a valid attack sequence).

We also assume the CVSS base metrics can be obtained for each exploit e as a vector bm of six numeric values, each of which corresponds to a base metric [12]. We will use the notation $bm[AV], bm[AC], \ldots, bm[A]$ to denote each corresponding element of the vector bm. Finally, we assume the dependency relationships between exploits are given using a function $adj()$, as formalized in Definition 1. That is, how the base metric of an exploit e is affected by another exploit e' will be reflected in the given value of $adj(e, e', m)$.

Definition 1: Given an attack graph G with the set of exploits E, define a function $adj() : E \times E \times \{AV, AC, Au, C, I, A\} \rightarrow [0, 1]$, and call $adj(e, e', m)$ the *adjusted value* for metric m of exploit e due to e'.

Next, we formalize the concept of *effective base metric* and *effective base score* in Definition 2. For each exploit e, the effective base metric simply takes the original base metric if no adjusted value is given. Otherwise, the effective base metric will take the highest adjusted value defined over any ancestor of e (note that an exploit may be affected by many exploits in different ways, leading to more than one adjusted values), because a metric should always reflect the worst case scenario (that is, the highest value) [2]. The effective base score basically applies the same equation to effective base metrics instead of the original metrics. In the definition, both effective base metric and score can be defined with respect to a given subset of exploits, which will be necessary later for the discussions in Section IV.

Definition 2: Given an attack graph G with the set of exploits E, the adjusted values given by function $adj()$, the CVSS base metric vector bm for each $e \in E$, and any $E' \subseteq E$ (E' will be omitted if $E' = E$), we define

- the *effective base metric* vector ebm of e with respect to E' as
  - $ebm[m] = bm[m]$, for each $m \in \{AV, AC, Au, C, I, A\}$, if $adj(e, e', m)$ is not defined for any ancestor e' of e in E'.
  - $ebm[m] = adj(e, e', m)$, if $adj(e, e', m)$ is the highest value defined over any ancestor e' of e in E'.
- the *effective base score* ebs of e as the base score calculated using Equation 1 with base metrics replaced by corresponding effective base metrics.

[1]Due to space limitations, we will only present the model and leave out algorithms for constructing the model, which are essentially modified versions of BN construction algorithms.

[2]This also explains how a *mutual dependency* between two exploits will be handled, that is, by taking the dependency that yields the higher aggregated value.

Approaches	Case 1	Case 2	Summary
Average	7.2	7.2	Ignoring causal relationships (exploiting one vulnerability enables the other)
Maximum	7.6	7.6	
Attack graph-based approach [18]	0.52	0.52	Ignoring dependency relationships (exploiting one vulnerability makes the other easier)
BN-Based approach [5]	0.61	0.52	Arbitrary adjustment for dependency relationships
Our approach	0.58	0.52	Adjustment with well-defined semantic

TABLE II
COMPARISON OF DIFFERENT APPROACHES

Definition 3 formalizes a Bayesian network (BN)-based model for aggregating the effective base scores. The directed graph is directly obtained from the attack graph. The conditional probabilities are assigned according to the causal relationships between an exploit and its pre- and post-conditions. Since the dependency relationships between exploits are already reflected in our definition of effective base scores, the BN needs not to explicitly model them. With the BN model, we can easily calculate the probability of satisfying any given goal conditions (or equivalently, the probability of important network assets being compromised).

Definition 3: Given attack graph G with exploits E, and the effective base score ebs for each $e \in E$, we define a Bayesian network $B = \langle G, Q \rangle$ where

- G is the attack graph interpreted as a directed graph with each vertex representing a random variable taking either T (true) or F (false), and the edges representing the direct dependencies among those variables.
- Q is the collection of conditional probabilities assigned as the following.
  - $P(c = T|e = T) = 1$, for each $e \in E$ satisfying $c \in post(e)$.
  - $P(e = T| \bigwedge_{\forall c \in pre(e)}(c = T)) = ebs/10$.

C. An Example

We now illustrate our approach by applying it to the example shown in Figure 3. The left-hand side shows a fictitious attack graph in which the dotted lines indicate dependency relationships, whose details will be given shortly. The right-hand side gives the corresponding model obtained by applying our formal framework as introduced above.

Specifically, we assume exploit B will give an attacker accesses to local network, which is required for exploiting D (since its base metric AV is $Local$), as indicated by the dotted line from B to D. This dependency relationship is modeled using the function $adj()$ on the right-hand side. Also, we assume that exploit C does not require an authenticated account (its base score Au is $None$), and exploiting C will give attackers the required account for exploiting D, as indicated by the dotted line from C to D, and modeled using the function $adj()$ on the right-hand side. Therefore, we can replace the base metric of exploit D with its effective base metric, as shown on the right-hand side, in order to calculate its effective base score as 8.77 (we assume the impact metrics to be $Complete$, $Complete$, and $Partial$). We then calculate $P(D = T)$ using the BN model shown in Figure 4 as $P(D = T) = \sum_{A,B,C \in \{T,F\}} P(D = T|B, C)P(C|A, B)P(A)P(B) = 0.27$.

A		B	
T	F	T	F
0.943	0.057	0.795	0.205

C			
A	B	T	F
T	F	0	1
F	F	0.877	0.123
T	T	0.877	0.123
F	T	0.877	0.123

D				
A	B	C	T	F
F	F	F	0	1
T	F	F	0	1
F	T	F	0	1
T	T	F	0	1
F	F	T	0.795	0.205
T	F	T	0.795	0.205
F	T	T	0.877	0.123
T	T	T	0.877	0.123

Fig. 4. The BN Model

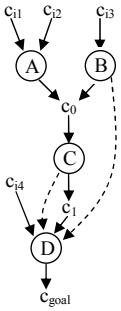
IV. THREE ASPECTS OF CVSS SCORES

We first demonstrate the need for interpreting and aggregating base scores from three aspects in Section IV-A. We then extend our approach in Section IV-B and illustrate it through an example in Section IV-C.

A. The Need for Different Aspects

The CVSS base metrics and scores can be interpreted in different ways. In this paper, we will consider three aspects of such metrics and scores (note that each metric or score may potentially be interpreted in one or more of those three aspects).

- First, as discussed in the previous section, the difference in scores may indicate different probabilities of attacks. For example, the numerical score assigned to the $AccessVector$ metric value of $AdjacentNetwork$ is lower than that of $Network$, which can be interpreted as that a vulnerability requiring local accesses has a lower *attack probability* than one that is remotely accessible.
- Second, we can also interpret the difference in scores as the minimum amount of *time and effort* by an average attacker. For example, a vulnerability



Adjusted Values:
$adj(D, C, AV) = 7.95$
$adj(D, B, Au) = 6.33$

Base Scores:

Exploits	AV	AC	Au	bs
A	Network	Low	None	9.43
B	Network	Medium	Single	7.95
C	Network	Medium	None	8.77
D	Local	Medium	Single	6

Effective base metric of D:
$ebm_D = \langle Network, Medium, None \rangle$

Effective base score of D:
$ebs_D = 8.77$

Fig. 3. An Example Attack Graph (Left) and The Corresponding Model (Right)

requiring multiple authentications at both OS and applications will likely demand more time and effort than one that requires no authentication.

- Third, the difference in scores may also imply the minimum *skill* level of an attacker who can successfully exploit that vulnerability. For example, exploiting a vulnerability with its $AccessComplexity$ score as $High$ will likely require more skills than exploiting one that has the value Low.

Interpreting the CVSS scores from different aspects will also require different methods for aggregating such scores. We demonstrate this fact through an example. Figure 5 shows a network consisted of four hosts (host 1 through 4) and another host on the internet (host 0). We assume there are firewalls between the hosts that prevent any traffic except those indicated by lines shown in the figure. We also assume host 1 through 4 each has a vulnerability, denoted by a letter inside parentheses. Finally, we assume the base scores are partially ordered, that is, vulnerability B is more severe than all others, and A is more severe than C (for simplicity, we do not explicitly distinguish between base scores and effective base scores in this example). We now consider how the scores should be aggregated for each of the three aspects.

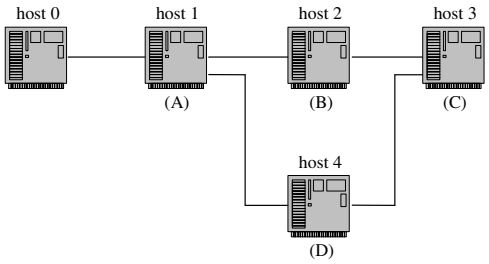


Fig. 5. An Example of Different Aspects

- First, suppose we have assigned probabilities P_A, P_B, P_C, and P_D to those four vulnerabilities based on the base scores. Also suppose host 3 is our main concern. The probability of host 3 being compromised can be calculated as $P = P_A * (P_B * P_C/(P_B+P_D)) * P_C$. Next, suppose we remove host 4 from the network. The probability will change to $P = P_A * P_B * P_C$, which is now smaller (i.e., host 3 is less likely to be compromised). This is reasonable since, by removing host 4, an attacker now has only one choice, that is, to reach host 3 via host 2. Finally, suppose we further remove host 2 from the network, the probability now becomes $P = P_A * P_C$, which is larger. This is also reasonable since an attacker now only needs to compromise host 1 before reaching host 3.
- Next, by considering a different aspect, the effort of an attacker, we can derive a different story. First, suppose we have assigned some effort scores F_A, F_B, F_C, and F_D to the four vulnerabilities based on their base scores (we will discuss how the effort scores should be defined later).
  Without considering dependency relationships, the effort spent on exploiting vulnerabilities will clearly be additive. Therefore, addition is the natural way to aggregate the effort scores. However, there is one more complication. In this example, an attacker may compromise host 3 in two ways, either through host 2 or host 4. Since a metric should measure the worst case scenario, it should yield the minimum effort required to compromise host 3. That is, $F = F_A + F_B + F_C$ (note that F_B is less than F_D due to our assumption).
  If we remove host 4 from the network, we can easily see that the effort score will remain the same, $F = F_A + F_B + F_C$, instead of becoming smaller, like in the above case of attack probability. This is reasonable since vulnerability D is not on the minimum-effort attack sequence so its removal will not affect the effort score. If we further remove host 2 from the network, we can see that the effort score now reduces to $F = F_A + F_C$.
- Finally, by considering the third aspect, the skill

level of an attacker, we can derive yet another story. First, suppose we have assigned some effort scores S_A, S_B, S_C, and S_D to the four vulnerabilities based on their base scores. Based on our assumption, we have that S_B is the smallest among the four and S_A is less than S_C. It is now easy to see that to compromise host 3, the minimum level of skills required for any attacker is S_C regardless of which sequence of attacks is being followed. Also, whether we remove host 4 or host 2 (or even host 1) from the network does not affect the skill score.

B. Aggregating Scores for Different Aspects

We now formalize our approach to aggregating scores for the effort and skill aspects. For both aspects, we will only consider the exploitability metric group, that is, the first three elements of the effective base metric vector.

In Definition 4, the formula and constants are merely designed to convert the exploitability score (defined in CVSS as the multiplication of the three metrics) to the same domain as the CVSS base score for consistency. The effective base metric vector of each exploit is now defined with respect to a given subset of exploits since whether a base metric needs to be adjusted will depend on which attack sequence is involved.

Definition 4: Given an attack graph G with the set of exploits E and the effective base metric vector ebm for each $e \in E$ with respect to some $E' \subseteq E$, we define for e both the *effort score* $es(e)$ and *skill score* $ss(e)$ with respect to E' as $\frac{0.6395}{ebm[AV]*ebm[AC]*ebm[Au]} - 0.2794$.

Although both scores are defined in the same way, they will need to be aggregated differently, as demonstrated in the previous section. Definition 5 formalizes the way we aggregate those scores. Roughly speaking, for aggregating effort score, we find an attack sequence whose summation of effort scores is the minimum among all attack sequences (although such an attack sequence is not necessarily unique, the minimum value is always unique); for aggregating skill scores, we find an attack sequence that whose maximum effort score is the minimum among all attack sequences.

Definition 5: Given an attack graph G with the set of exploits E, and the effort score $es(e)$ and skill score $ss(e)$ for each $e \in E$, we define

- the *accumulative effort score* of e as $F(e) = min(\{\sum_{e' \in q} es(e') : q$ is an attack sequence with e as its last element$\})$ (here $es(e')$ is defined with respect to q).
- the *accumulative skill score* of e as $S(e) = min(\{max(\{ss(e') : e' \in q\}) : q$ is an attack sequence with e as its last element$\})$ (here $ss(e')$ defined with respect to q).

C. An Example

Now we demonstrate how our approach can be applied to calculate the accumulative effort and skill scores through a more elaborated example. The left-hand side of Figure 6 shows an example attack graph in which two attack sequences can both lead to the assumed goal condition. In the upper right-hand side we show CVSS metrics of the vulnerabilities. The dashed lines in the attack graph indicate dependency relationships between the exploits. Specifically, the adjusted $AccessVector$ metric value of C should be $Network$ and the adjusted $Authentication$ metric value of F should be $None$.

The calculated cumulative effort scores and cumulative skill scores are shown on the lower right-hand side. Note that in calculating the scores for each sequence, we need the effort and skill scores that are defined with respect to that sequence. In particular, exploit F has two different effort and skill scores, since its effective base metric $Authentication$ is adjusted in sequence q_2 (due to exploit E) but not in sequence q_1.

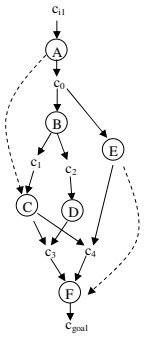
V. Simulation

This section presents simulation results [3]. Our objective is to compare our approach to existing ones through numerical results and to examine how close those results are to the statistically expected results represented by simulated attackers. To the best of our knowledge, the simulation presented here is among the first efforts on experimentally evaluating network security metrics.

We employ the Boston university Representative Internet Topology gEnerator(BRITE) [3] to generate simulated network topologies. Vulnerability information is injected into the generated network topologies to obtain network configurations. Finally, attack graphs are generated from the configurations using the standard two-pass procedure [1]. All simulations are conducted on a computer equipped with a 3.0GHz CPU and 8GB RAM.

The objective of the first two simulations is to evaluate our approach from the aspect of attack probability, as detailed in Section III. For simplicity, we assign random base metrics to vulnerabilities, and dependency relationships to pairs of vulnerabilities, while leaving more realistic approaches for future work. We then apply both our approach and the existing BN-based approach by Frigault *et al.* [5] to calculate the probability of attacks with respect to a set of randomly chosen goal conditions. We also compare our results to the percentage of simulated attackers (each of which is modeled as a

[3]To the best of our knowledge, there do not exist public datasets that contain a sufficient number of real world attack graphs which can be used for experiments.



	AV	AC	Au	es, ss
v_A	Network	Low	None	1
v_B	Network	Medium	None	1.21
v_C	Local	Low	None	1 (w.r.t. q_1)
v_D	Local	Medium	None	3.49
v_E	Network	Medium	Single	1.59
v_F	Network	Medium	Single	1.59 (w.r.t. q_1) and 1.21 (w.r.t. q_2)

Attack Sequence	Effort $F(F)$	Skill $S(F)$
$q_1 : A \rightarrow B \rightarrow C \rightarrow F$	4.8	1.59
$q_2 : A \rightarrow B \rightarrow D \rightarrow E \rightarrow F$	8.5	3.49

Fig. 6. An Example Attack Graph (Left) and The Effort and Skill Scores (Right)

random subset of exploitable vulnerabilities) who can successfully reach the goal conditions.

In Figure 7, the X-axis is the average effective base score of all vulnerabilities in each network divided by 10, denoted by β. The Y-axis is either the aggregated score of attack probability (for both our approach and the approach by Frigault *et al.*) or the percentage of successful attackers. Each result is the average of 500 simulations on different network configurations. The curve $Simulation$ corresponds to the simulated attackers, which is used as a baseline for comparison. The line β corresponds to the naive approach of taking the average base score among all vulnerabilities in a network, which is clearly inaccurate.

In Figure 7, the curve S_0 corresponds to our approach and the curve S_1 the approach by Frigault *et al.*. Clearly, our result is closer to the simulated attackers than theirs. Also, our probability is always higher than theirs due to the proper handling of dependency relationships. In Figure 7, we have assigned dependency relationships to n pairs of randomly chosen vulnerabilities where n is drawn from a uniform distribution on $[0, 3]$. Figure 8 shows a similar simulation, except that we increase the amount of dependency relationships to n pairs where n is now drawn from a (uniform distribution on $[0, 5]$. The results show that our approach is still very close to the simulated attackers, whereas Frigault's result further deviates from the baseline results.

The objective of the next simulation is to study the deviation of aggregated scores from the baseline of simulated attackers. For this purpose, Figure 9 depicts the results computed on 800 different networks. The X-axis is the percentage of simulated attackers who can reach the goal conditions, and the Y-axis is the aggregated probability score. The dots S_0 and S_1 correspond to the results of our approach and Frigault's, respectively. The two solid lines labeled with S_0 and S_1 represent the average probability score within each 0.05 interval of the X-axis. The two polygon areas depict the distribution of aggregated scores produced by the two approaches. As we can see from the figure, our results evenly spread around the simulated attackers' results (represented by the diagonal line), whereas Frigault's results are almost always lower.

The next simulation aims to evaluate our approach from the skill aspect. For this purpose, each simulated attacker is randomly assigned a skill level based on exponential distribution (significantly less attackers possess a higher level of skills). Each simulated attacker can only exploit those vulnerabilities whose skill scores (as defined in Section IV) are no greater than the attacker's assigned skill level. In Figure 10, the X-axis is the percentage of successful simulated attackers, and the Y-axis is either the skill score produced by our approach or the skill level of simulated attackers. Each result is the average of 100 simulations. The curve *Skill metric* is the cumulative skill score of our approach; the curve *Minimal skill* corresponds to the lowest skill level of simulated attackers among those who can reach the goal conditions. We can see that those two curves almost overlap each other, indicating the accuracy of our approach. The curve *Average skill* shows the average skill level among successful simulated attackers, which has the same trend, but is always higher than our result. The curve *Vulnerability average* shows the average skill score of all vulnerabilities to be not so good a metric.

The last simulation evaluates our approach from the effort aspect. For this purpose, each simulated attacker is randomly assigned an*effort threshold* based on exponential distribution (less attackers are willing to spend more effort). We assume each simulated attacker will only exploit those vulnerabilities whose effort scores (as defined in Section IV) are no greater than the attacker's assigned effort threshold. In Figure 11, the X-axis is the percentage of successful simulated attackers, and the Y-axis is either the effort score or the effort threshold (of simulated attackers). The curve *Effort metric* is the cumulative effort score of our approach; the curve *Minimal effort* and *Average effort* respectively correspond to the lowest and average effort threshold of those simulated attackers who successfully reach the goal conditions.

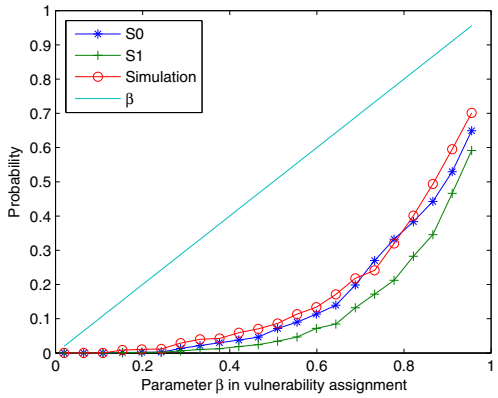


Fig. 7. The Probability Aspect

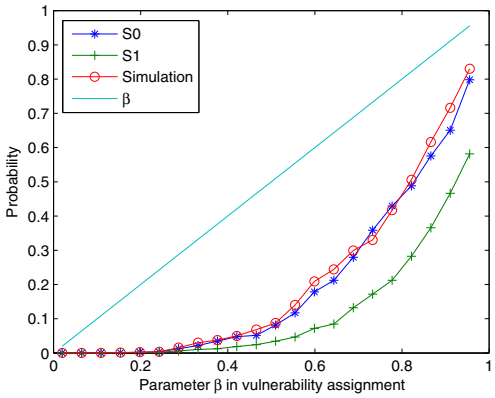


Fig. 8. Increased Dependency Relationships

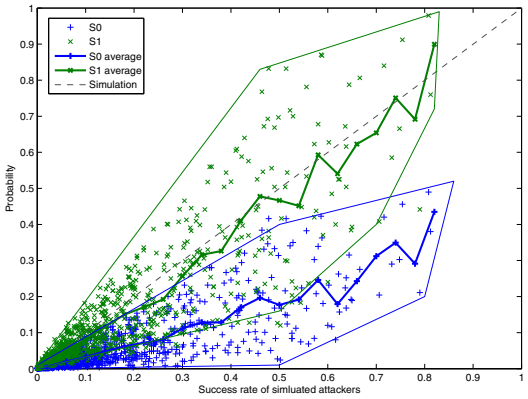


Fig. 9. Distribution of Probability Scores

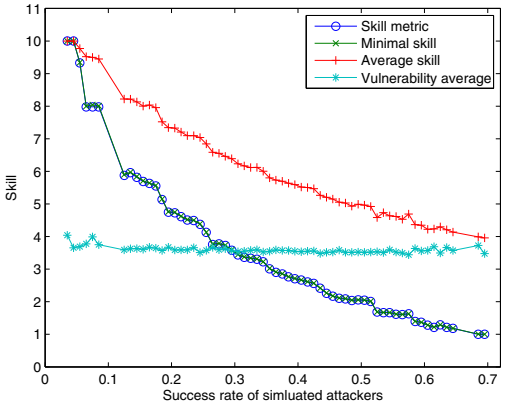


Fig. 10. The Skill Aspect

Again, we can see our effort scores closely match the minimum required effort and follow the same trend as the average effort.

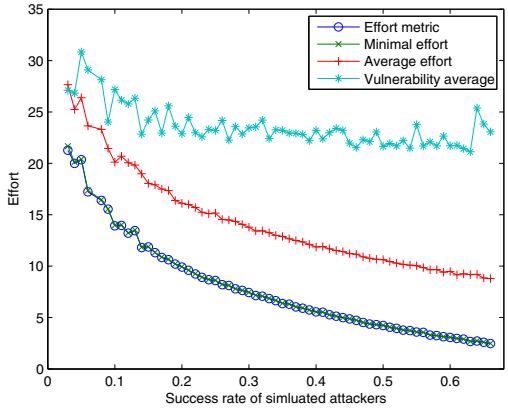


Fig. 11. The Effort Aspect

VI. Related Work

Existing standardization efforts on security metrics include the Common Vulnerability Scoring System (CVSS) [12] and more recently the Common Weakness Scoring System (CWSS) [17]. Both CVSS and CWSS measure the relative severity of individual vulnerabilities in isolation and do not address their overall impact. The research on security metrics has attracted much attention [8], [15]. In a recent work [11], the authors rank states in an attack graph based on probabilities of attackers reaching these states during a random simulation. In [2], the amount of minimum effort needed along such paths is used as a metric. A similar work replaces attack trees with more advanced attack graphs and replace attack paths with attack scenarios [14]. A Mean Time-to-Compromise metric is proposed based on the predator state-space model (SSM) used in the biological sciences in [9]. In our recent work, we have proposed a general framework for designing network security metrics [22], an attack resistance-based metric [23], Bayesian network-based metrics [5], [4], a probabilistic

approach [18], and a quantitative method for hardening networks [21]. Parallel to our work on probabilistic security metrics, the authors in [6] address several important issues in calculating such metrics including the dependencies between different attack sequences in an attack graph and cyclic structures in such graphs.

In this paper, we focus on known vulnerabilities whose CVSS scores are readily available. Most existing work focus on developing security metrics for known vulnerabilities in a network. A few exceptions include an empirical study on the total number of zero day vulnerabilities available on a single day based on existing facts about vulnerabilities [10], and an empirical study on software vulnerabilities' life cycles [16]. Another recent effort ranks different applications in the same system by how serious the consequence would be if there exists a single zero day vulnerability in those applications [7].

In this paper, we borrow the compact attack graph model given in [1] while incorporating security metric scores. Attack graphs have also been applied to a wide range of other security applications, such as alert correlation and prediction [19], [20].

VII. Conclusion

In this paper, we have addressed two important limitations of existing approaches to aggregating CVSS scores, namely, the loss of useful semantics in handling dependency relationships and the loss of semantics from different aspects. We have proposed the novel approaches of base metric-level aggregation and aggregating from three different aspects. The simulation results confirmed the advantages of our approach. Future work will be directed to incorporating temporal and environmental scores, considering other aspects for interpreting the scores, and experiments with more realistic settings.

Acknowledgements The authors thank the anonymous reviewers for their valuable comments. This material is based upon work supported by National Institute of Standards and Technology Computer Security Division under grant 70NANB11H126, by the US Army Research Office under MURI grant W911NF-09-1-0525 and DURIP grant W911NF-11-1-0340, by the Air Force Office of Scientific Research under grant FA9550-09-1-0421, and by Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations.

References

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of ACM CCS'02*, 2002.

[2] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 1st ACM QoP*, 2005.

[3] Boston university representative internet topology generator. Available at http://www.cs.bu.edu/brite/.

[4] M. Frigault and L. Wang. Measuring network security using bayesian network-based attack graphs. In *Proceedings of The 3rd IEEE International Workshop on Security, Trust, and Privacy for Software Applications (STPSA'08)*, 2008.

[5] M. Frigault, L. Wang, A. Singhal, and S. Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of 4th ACM QoP*, 2008.

[6] J. Homer, X. Ou, and D. Schmidt. A sound and practical approach to quantifying security risk in enterprise networks. Technical Report, 2009.

[7] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *Proceedings of ACSAC'09*, pages 117–126, 2009.

[8] A. Jaquith. *Security Merics: Replacing Fear Uncertainity and Doubt*. Addison Wesley, 2007.

[9] D. Leversage and E. Byres. Estimating a system's mean time-to-compromise. *IEEE Security and Privacy*, 6(1):52–60, 2008.

[10] M. McQueen, T. McQueen, W. Boyer, and M. Chaffin. Empirical estimates and observations of 0day vulnerabilities. *Hawaii International Conference on System Sciences*, 0:1–12, 2009.

[11] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing. Ranking attack graphs. In *Recent Advances in Intrusion Detection 2006*, 2006.

[12] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6):85–89, 2006.

[13] National vulnerability database. available at: http://www.nvd.org, May 9, 2008.

[14] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the ACM QoP*, pages 31–38, 2006.

[15] R. Savola. Towards a taxonomy for information security metrics. In *Proceedings of the 3rd ACM QoP*, pages 28–30. ACM, 2007.

[16] M. Shahzad, M. Shafiq, and A. Liu. A large scale exploratory analysis of software vulnerability life cycles. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*, 2012.

[17] The MITRE Corporation. Common weakness scoring system. http://cwe.mitre.org/cwss/, 2010.

[18] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *Proceedings of the 22nd IFIP DBSec*, 2008.

[19] L. Wang, A. Liu, and S. Jajodia. An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, pages 247–266, 2005.

[20] L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, 2006.

[21] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 11 2006.

[22] L. Wang, A. Singhal, and S. Jajodia. Measuring network security using attack graphs. In *Proceedings of the 3rd ACM QoP*, New York, NY, USA, 2007. ACM Press.

[23] L. Wang, A. Singhal, and S. Jajodia. Measuring the overall security of network configurations using attack graphs. In *Proceedings of 21th IFIP DBSec*, 2007.