Measurements of the Most Significant Software Security Weaknesses

Carlos Cardoso Galhardo
National Institute of Standards and Technology; INMETRO
carlos.cardosogalhardo@nist.gov;cegalhardo@inmetro.gov.br

Peter Mell
National Institute of Standards and Technology
peter.mell@nist.gov

Irena Bojanova
National Institute of Standards and Technology
irena.bojanova@nist.gov

Assane Gueye
UADB-Senegal & Prometheus Computing
assane1.gueye@uadb.edu.sn

ABSTRACT

In this work, we provide a metric to calculate the most significant software security weaknesses as defined by an aggregate metric of the frequency, exploitability, and impact of related vulnerabilities. The Common Weakness Enumeration (CWE) is a well-known and used list of software security weaknesses. The CWE community publishes such an aggregate metric to calculate the 'Most Dangerous Software Errors'. However, we find that the published equation highly biases frequency and almost ignores exploitability and impact in generating top lists of varying sizes. This is due to the differences in the distributions of the component metric values. To mitigate this, we linearize the frequency distribution using a double log function. We then propose a variety of other improvements, provide top lists of the most significant CWEs for 2019, and provide an analysis of the identified software security weaknesses.

CCS CONCEPTS

• **Security and privacy → Vulnerability management**; **Software and application security**.

KEYWORDS

Security, Weakness, Software Flaw, Severity

ACM Reference Format:
Carlos Cardoso Galhardo, Peter Mell, Irena Bojanova, and Assane Gueye. 2020. Measurements of the Most Significant Software Security Weaknesses . In *Proceedings of Measurements of the Most Significant Software Security Weaknesses (Most Significant Software Security Weaknesses).* ACM, New York, NY, USA, Article ?, 11 pages. https://doi.org/??

1 INTRODUCTION

In 2019, there were over 17 000 documented software vulnerabilities [21] that enable malicious activity. While many are discovered, they map to a relatively small set of underlying weakness types. We posit that if the most significant of these types can be identified, developers of programming languages, software, and security tools can focus on preventing them and thus over time diminish the quantity and severity of newly discovered vulnerabilities.

Most Significant Software Security Weaknesses, 12 2020, Austin, USA

ACM ISBN ??…$15.00
https://doi.org/??

In this work, we provide a metric to calculate the most significant security weaknesses (MSSW) in software systems. We define a 'significant' weakness as one that is both frequently occurring among the set of publicly published vulnerabilities and results in high severity vulnerabilities (those that are easily exploitable and have high impact). The set of security weakness types upon which we calculate significance comes from the Common Weakness Enumeration (CWE) [14]. We also leverage the Common Vulnerabilities and Exposures (CVE) [12] repository of publicly announced vulnerabilities, the Common Vulnerability Scoring System (CVSS) [6] to measure the severity of vulnerabilities, and the National Vulnerability Database (NVD) [21] to map the CVEs to both CWEs and CVSS scores.

In the fall of 2019, the CWE community published an equation to calculate the 'Top 25 Most Dangerous Software Errors' (MDSE) among the set of CWEs [16]. The MDSE equation claims to combine 'the frequency that a CWE is the root cause of a vulnerability with the projected severity'. However, we empirically find that the equation highly biases frequency and almost ignores severity in generating top lists of varying sizes. This is due to the equation multiplying calculated frequency and severity values together while each of them having very different distributions. Frequency distributions have a power law like curve, while severity distributions are more uniform. Our mitigation is to create a revised equation, named MSSW, that adjusts the frequency distribution using a double log function to better match it to the severity distribution. We also fix an error in how normalization is done in the MDSE equation.

We next improve upon the data collection approach used by the MDSE equation by leveraging published literature. Lastly, we publish top lists of the most significant CWEs for 2019 and provide a brief analysis of those software security weaknesses. It is our hope that this data and our methodology will be adopted to focus our collective security resources in reducing the most significant software security weaknesses.

The rest of this work is organized as follows. Section 2 provides background on CVE, CVSS, CWE, NVD, and the MDSE equation. Section 3 discusses the limitations of the MDSE equation. Section 4 presents our MSSW equation that mitigates the previously identified limitations. Section 5 provides two lists of the most significant CWEs at two different levels of software flaw type abstractions. Section 6 provides a discussion and analysis of the most significant CWEs identified. Section 7 presents related work and Section 8 concludes.

2 BACKGROUND

2.1 Common Vulnerabilities and Exposures

The CVEs are a large set of publicly disclosed vulnerabilities in widely-used software. They are enumerated with a unique identifier, described, and referenced with external advisories [12] [1].

2.2 Common Vulnerability Scoring System

CVSS 'provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity' [5]. The CVSS base score reflects the inherent risk of a vulnerability apart from any specific environment. The base score is composed from two sub-scores that calculate exploitability (how easy it is to use the vulnerability in an attack) and impact (how much damage the vulnerability can cause to an affected component).

The exploitability score is determined by the following:

- attack vector: 'the context by which vulnerability exploitation is possible',
- attack complexity: 'the conditions beyond the attacker's control that must exist in order to exploit the vulnerability',
- privileges required: 'the level of privileges an attacker must possess before successfully exploiting the vulnerability', and
- user interaction: a human victim must participate for the vulnerability to be exploited

The impact score is determined by measuring the impact to the confidentiality, integrity, and availability of the affected system. Also included is a scope metric that 'captures whether a vulnerability in one vulnerable component impacts resources in components beyond its security scope'. The specifics on these metrics and the details for the three equations can be found in the CVSS version 3.1 specification at [6].

2.3 Common Weakness Enumeration

The Common Weakness Enumeration (CWE) [9] is a 'community-developed list of common software security weaknesses'. 'It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts' [14]. It contains an enumeration, descriptions, and references for 839 software weaknesses that are referred to as CWEs, where each is labelled CWE-X with X being an integer.

The CWE weaknesses model has four layers of abstraction: pillar, class, base, and variant. There is also the notion of a compound, that associates two or more interacting or co-occurring CWEs [17]. These abstractions reflect to what extent issues are described in terms of five dimensions: behavior, property, technology, language, and resource. Variant weaknesses are at the most specific level of abstraction; they describe at least three dimensions. Base weaknesses are more abstract than variants and more specific than classes; they describe two to three dimensions. Class weaknesses are very abstract; they describe one to two dimensions, typically not specific about any language or technology. Pillar weaknesses are the highest level of abstraction.

There are a set of taxonomies, called views, to help organize the CWEs. Two prominent CWE taxonomies are the 'Research Concepts' (view 1000) and 'Development Concepts' (view 699). There is also a view 1003 that was made specifically to describe the set of CVEs that contains 124 CWEs. It is called 'CWE Weaknesses for Simplified Mapping of Published Vulnerabilities View'.

2.4 National Vulnerability Database

The CWE effort uses the National Vulnerability Database (NVD) [21] as a repository of data from which to calculate the MDSE scores. The NVD contains all CVEs and for each CVE it provides a CVSS score along with the applicable CWE(s) that describe the weakness(es) enabling the vulnerability. For the empirical work in this paper, we use the complete set of 17 308 CVEs published by NVD for 2019, that were available as of 2020-03-19.

2.5 Most Dangerous Software Error Equation

The MDSE equation is designed to balance the frequency and severity in ranking the CWEs. The frequency is determined by the number of CVEs that map to a given CWE in the time period of study. The severity is determined by the mean CVSS score for the CVEs mapped to a given CWE. The MDSE score for a CWE is produced by multiplying the normalized frequency by the normalized severity and then multiplying by 100. We now describe this metric more formally.

2.5.1 Metric for Normalized Frequency. Let I designate the set of all CWEs and let J be the set of all CVEs.

For CWE $i \in I$, let N_i be the number of CVEs mapped to i, defined as follows:

$$N_i = \sum_{j \in J} e_{ij}, \tag{1}$$

where

$$e_{ij} = \begin{cases} 1, & \text{if CVE } j \text{ is mapped to CWE } i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Now let F_i be the normalized frequency for CWE i, defined as follows:

$$F_i = \frac{N_i - \min\limits_{i' \in I}(N_{i'})}{\max\limits_{i' \in I}(N_{i'}) - \min\limits_{i' \in I}(N_{i'})}. \tag{3}$$

2.5.2 Metric for Normalized Severity. Let J, N_i, and e_{ij} be as defined above in Section 2.5.1. Let s_j be the CVSS base score for CVE j. For CWE $i \in I$, let $\overline{S_i}$ be the mean CVSS score, defined as follows:

$$\overline{S_i} = \frac{\sum_{j \in J} s_j e_{ij}}{N_i}. \tag{4}$$

Now let S_i be the normalized severity for CWE i, defined as follows:

$$S_i = \frac{\overline{S_i} - \min\limits_{j \in J}(s_j)}{\max\limits_{j \in J}(s_j) - \min\limits_{j \in J}(s_j)}. \tag{5}$$

2.5.3 Most Dangerous Software Error Metric. Let $MDSE_i$ be the MDSE score for CWE i, defined as follows:

$$MDSE_i = F_i * S_i * 100. \tag{6}$$

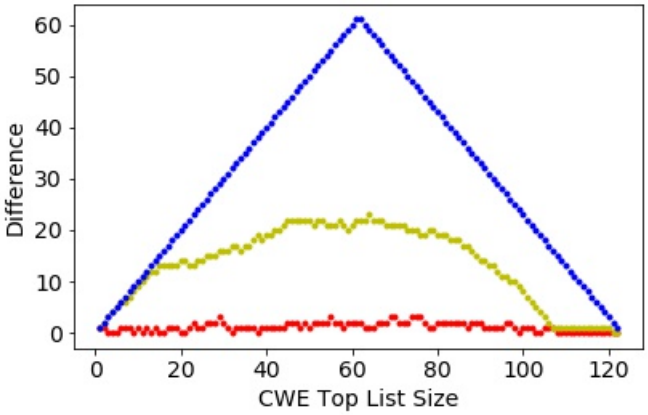

Figure 1: The Size of the Set Difference between Top Lists from the MDSE Equation Compared to Frequency Top Lists (red bottom line), Severity Top Lists (yellow middle line), and the Theoretical Maximum (blue top line)

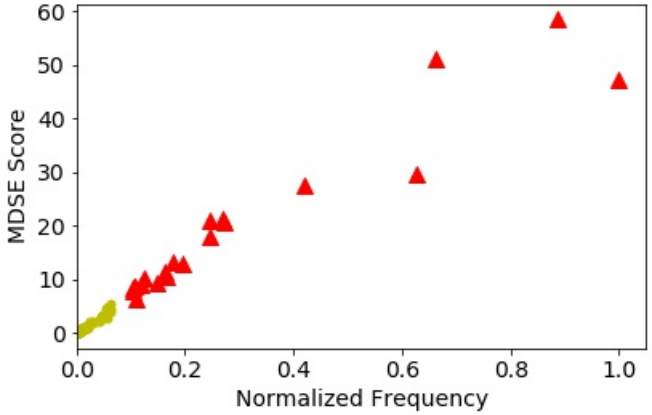

Figure 2: CWEs Chosen (Red Triangles) and Not Chosen (Yellow Circles) for a MDSE Top 20 List Relative to Frequency

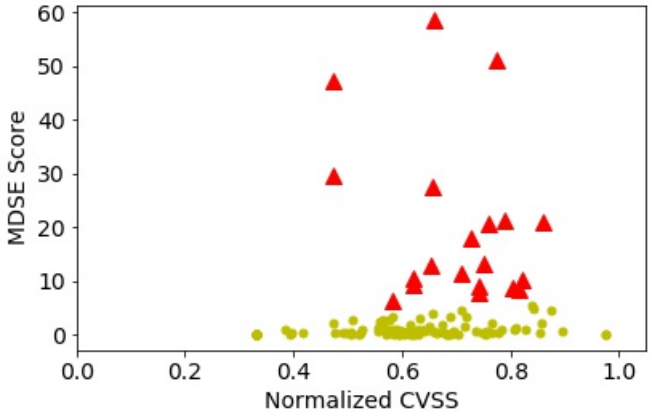

Figure 3: CWEs Chosen (Red Triangles) and Not Chosen (Yellow Circles) for a MDSE Top 20 List Relative to Severity

3 LIMITATIONS OF THE EQUATION

The MDSE equation was designed to and appears to combine both frequency and severity in determining the individual scores used to rank the CWEs. The frequency component is calculated in equation 3 and the severity component is calculated in equation 5; both are brought together in equal proportions in equation 6 to create the MDSE score. And both the severity and frequency are normalized in equations 3 and 5 to ensure that their scales match for the multiplication in equation 6.

However, we empirically find that the MDSE equation strongly biases frequency over severity. To demonstrate this, we calculate MDSE top CWE lists for all possible list sizes. While there exist 839 CWEs, the CVE data used as MDSE input is mapped only to 124 view 1003 CWEs (see section 2.3)[1]. Thus the maximum top list size is 124. We also calculate top CWE lists using just the frequency equation 3 and then just the severity equation 5. For each CWE top list size, we perform a set difference between the MDSE top list and the frequency top list. We then also do this between the MDSE top list and the severity top list. The size of the set difference between the MDSE top list and the frequency top list (for all possible top list sizes) has a maximum difference of 3. The size of the set difference between the MDSE top list and the severity top list (for all possible top list sizes) has a maximum difference of 23. This is shown graphically in Figure 1. The bottom red line represents the set difference using frequency and the yellow middle line represents the set difference using severity. The top blue line shows the maximal possible set difference that could be achieved using the 124 CWEs. Note how the bottom red line indicates that the top frequency list and the top MDSE list are almost identical for all CWE top list sizes. The middle yellow line shows how the top severity list and the top MDSE list deviate dramatically; they have almost a maximal difference for top list sizes of up to 15.

[1]This is expected as view 1003 was designed to cover the types of vulnerabilities in CVE.

3.1 Limitation 1: Distribution Differences

The MDSE equation in practice biases frequency over severity, even though its equations treat them equally, because frequency and severity have very different distributions. The frequency distribution has the majority of CWEs at a very low frequency and a few at a very high frequency (somewhat resembling a power law curve). This can be seen in Figure 2 by looking at how each CWE maps to the x-axis (note that most of the yellow dots overlap, there are 102 yellow dots and 20 red triangles). The figure shows the MDSE scores for each CWE and shows how (for a top list of size 20) the top scoring chosen CWEs are exactly the most frequent CWEs. This is not unique and occurs for many top lists (e.g., for sizes 11, 13, 15, 16, 20, 21, 32, and 38) as shown when the bottom red line is at 0 in Figure 1. The other sizes of top lists produce graphs that are almost identical to that in Figure 2, with at most 3 yellow circles just to the right of the leftmost red triangles representing the chosen CWEs.

The severity distribution is more uniform within a limited range. It can be seen in Figure 3 by looking at how the CWEs map to the x-axis. This figure shows how the top MDSE scoring chosen CWEs do not necessarily map to the CWEs with the highest severity. In fact, only 1 of the top 10 most severe CWEs made the MDSE top 20 list (note that many of the yellow circles lay on top of each other).

3.2 Limitation 2: Normalization Error

Equation 5 normalizes S_i based on the maximum and minimum CVSS score found in the set of inputted CVEs. However, this does not lead to the expected and desired normalized distribution from 0 to 1. For our data the range is from .28 to .97, as can be seen from the mappings of the points onto the x-axis in Figure 3. The reason for this is that $\overline{S_i}$ has a smaller range than the maximum and minimum CVSS score because each $\overline{S_i}$ represents the mean of the CVSS score for the CVEs that map to CWE i. This limitation, while of less consequence than the previous, constrains the range of S_i values thus further lessening the influence that severity has in determining a MDSE score.

4 MITIGATED EQUATION

We mitigate the limitations of the MDSE equation by replacing equations 3, 5, and 6 with the five equations that follow:

$$k = \frac{1}{\log_e \log_e \max_{i \in I}(N_i)}, \tag{7}$$

$$F_i' = \begin{cases} \log_e N_i, & \text{if } N_i >= 1, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

$$F_i'' = \begin{cases} k \log_e F_i', & \text{if } F_i' >= 1, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

$$S_i' = \frac{\overline{S_i} - \min_{i' \in I}(\overline{S_{i'}})}{\max_{i' \in I}(\overline{S_{i'}}) - \min_{i' \in I}(\overline{S_{i'}})}, \tag{10}$$

$$MSSW_i = F_i'' * S_i' * 100. \tag{11}$$

4.1 Explanation of Mitigated Equation

Equation 8 takes the log of the frequency using the natural log as the base. Equation 9 then takes the log of equation 8, again using the natural log as a base and multiplies the result by k (from equation 7). The k coefficient serves the purpose of normalizing the resulting values between 0 and 1 (to match the severity range in equation 10).

These three equations modify the power law like frequency distribution to make it more linear, thus addressing limitation 1 (from Section 3.1). This can be seen in Figure 4. Each value on the x-axis represents a particular CWE, ordered from least frequent to most frequent. The lower blue line represents the normalized frequency (i.e., number of CVEs mapped to a particular CWE). Note the slow increase in frequency up to the 100th CWE, followed by a rapid increase terminating in an almost vertical line. The middle yellow line represents taking the log of the frequency (equations not shown) which helps linearize but still results in an upwards curve on the right side. Thus, we apply a double log for further linearization (see the top red line). We note that this approach is not pseudo-linear for the most infrequent of CWEs. However, this does not cause problems as our goal is to identify the most significant and any such CWE must have at least a moderate frequency.

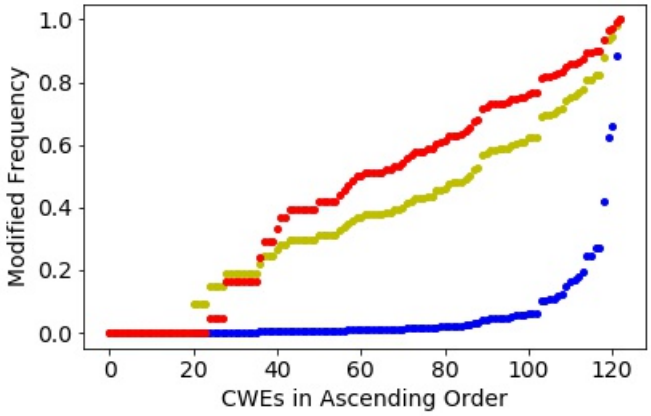

Figure 4: Normalized Distributions of Frequency (bottom blue line), Log of Frequency (middle yellow line), and Double Log of Frequency (top red line).

Our modified MDSE equation 11 then multiplies frequency and severity as in the original MDSE equation, but it multiplies from two distributions that have a similar shape for the part of the functions that are of interest. This enables the MSSW equation to more fairly balance evaluating frequency and severity in scoring and ranking a CWE.

To address limitation 2 from Section 3.2, equation 10 normalizes the severity using the maximum and minimum mean severity values. This gives the distribution a full 0 to 1 range which is not achieved in the MDSE equation 5.

Equation 11 is our final modified MDSE equation. We recommend its use in place of the published MDSE equation.

4.2 Analysis of Mitigated Equation

We now conduct three experiments to evaluate the effect of the MSSW equation in making the frequency and severity distributions more similar and in producing top lists with more equal inclusion of both frequency and severity. A fourth experiment involving correlation calculations is provided in Section 5 (because it includes some variants introduced in that section).

4.2.1 Risk Map Experiment. Figure 5 shows an MDSE risk map for the evaluated CWEs. Each red dot represents a CWE positioned according to its S_i severity and F_i frequency. In general, CWEs towards the upper right are more significant and those towards the lower left are less significant. Note how the majority of the CWEs are squished very close to the x-axis as many have a very small frequency. Also, the range of x-values is constrained from .37 to .97 (when the normalization should make it from 0 to 1).

Figure 6 shows the same risk map using our double log frequency F_i'' and our modified severity S_i'. Note how the CWEs are now more uniformly spread over the y-axis. Also, the range of x-axis values is

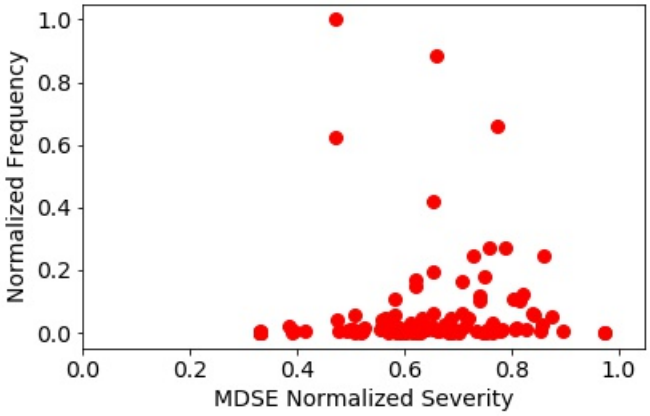

Figure 5: MDSE Equation Risk Map

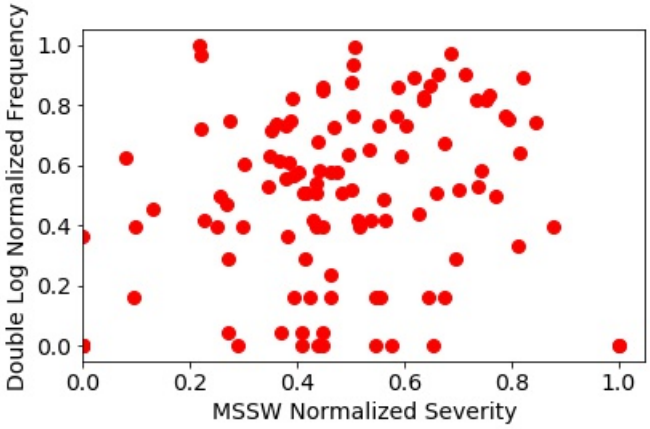

Figure 6: MSSW Equation Risk Map

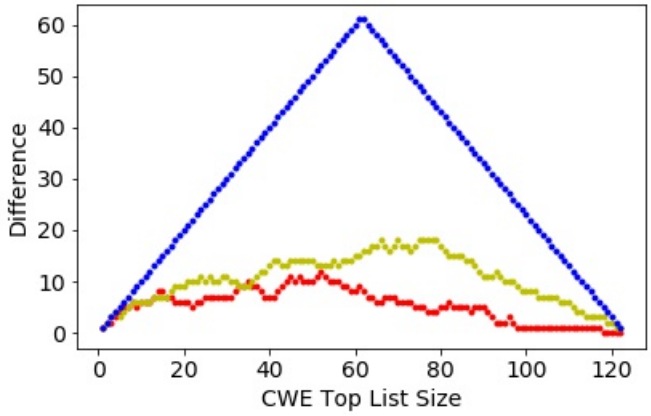

Figure 7: The Size of the Set Difference between Top Lists from the MSSW Equation Compared to Frequency Top Lists (red lower line), Severity Top Lists (yellow middle line), and the Theoretical Maximum (blue top line)

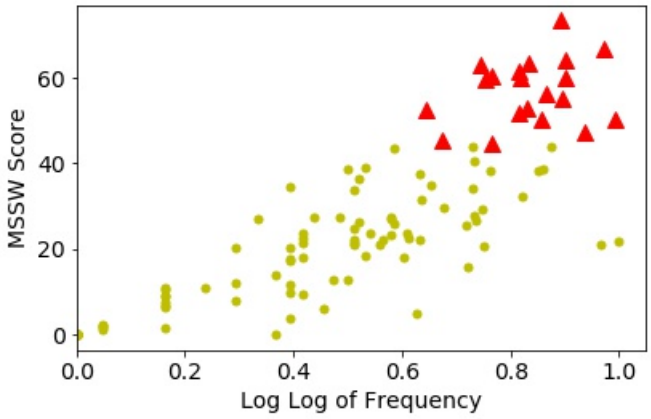

Figure 8: CWEs Chosen (Red Triangles) and Not Chosen (Yellow Circles) for a MSSW Top 20 List Relative to Frequency

now from 0 to 1. The MSSW equation that combines frequency and severity using the values shown in Figure 6 will now more equally combine them than with the MDSE values shown in Figure 5.

4.2.2 Set Difference Experiment. In Figure 7 we show the size of the set difference between the MSSW top list and the severity top list (the mostly lower red line). We also calculate the set difference between the MSSW top list and the frequency top list (the middle yellow line). Note how the red and yellow lines are much closer together than in Figure 1 and how the red line does not hover close to 0 like it does in Figure 1. This demonstrates that the MSSW equation is more evenly balancing inclusion of the top frequency and top severity CWEs.

Note that the goal is not to have the red and yellow lines match. The top list should not necessarily evenly include an equal number of both top frequency and top severity CWEs. Our point with this analysis is to show how the MDSE equation almost exclusively chooses the top frequency CWEs and how our MSSW equation factors in CWEs from both sets. The next subsection will evaluate this more equal inclusion in more detail, focusing on top lists of size 20.

4.2.3 Chosen CWE Experiment. Figure 8 shows the MSSW scores plotted against the double log frequency F_i'' scores. Each point represents a CWE. The red triangles indicate the CWEs that were chosen for the MSSW top 20 list. Note how unlike in the analogous Figure 2 for MDSE, there are many higher frequency CWEs that are not chosen for the top 20 list due to their severity not being high enough.

Likewise, Figure 9 shows the MSSW scores plotted against the S_i' normalized mean CVSS score for each CWE. Note how the range spreads from 0 to 1, unlike the analogous Figure 3 for the MDSE equation. Also note how the MSSW equation chooses CWEs for the top 20 list from CWEs with generally higher CVSS scores. However,

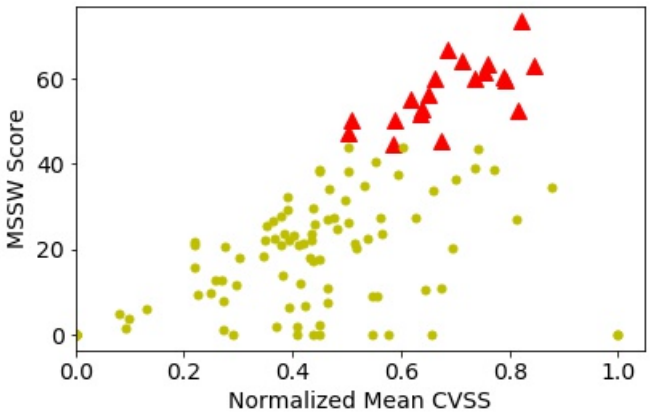

Figure 9: CWEs Chosen (Red Triangles) and Not Chosen (Yellow Circles) for a MSSW Top 20 List Relative to Severity

it excludes many high severity CWEs because their frequencies were too low.

5 2019 TOP 20 LISTS OF THE MOST SIGNIFICANT WEAKNESSES

We now use our MSSW equation to generate lists of the most significant software security weaknesses. We follow the approach in [11] of separately providing a top list for CWEs of higher levels of abstraction (pillars and classes) apart from a list covering CWEs of lower levels of abstraction (bases, variants, and compounds). We will refer to the higher level abstraction list as the class list and the lower level abstraction list as the base list for convenience and because both lists are primarily composed of either classes or bases. We also follow [11] in using published CWE taxonomy views 1000 and 1008 (discussed in Section 2.3) to propagate CVE data from child CWEs to their parents. This provides a more accurate mapping of CVEs onto the CWEs, providing a more accurate data foundation upon which to apply our MSSW equation.[2]

These modifications also alter the frequency and severity distributions which could potentially render our double log function invalid. However, Table 1 shows correlation results for using and not using all combinations of the modifications adopted from [11]. It shows that the MDSE equation is highly correlated to frequency (.97 or higher) with very little correlation to severity (.25 or lower) regardless of the modifications used or not used. It also shows that the MSSW equation is strongly correlated to both frequency (.81 or higher with one exception) and severity (.66 or higher) regardless of the modifications used. Our one exception is for the class list using propagation with MSSW; even here the frequency correlation was .55 (still strong but much less than the others).

[2]This propagation especially helps the formulation of the class list since most classes have children. It has a lesser effect on the base list. Note that it is impossible to inverse the propagation of data. CWEs are labelled as specifically as possible by NVD analysts so CVEs described by pillar or class CWEs do not get reflected in the base list. It is even possible that they shouldn't because there may be unidentified bases missing from view 1003 that are still covered by the view 1003 classes.

Table 1: Measurements Showing the Pearson Correlation of MDSE and MSSW to Frequency and Severity

			Correlation	
Equation	Abstraction	Propagation	Frequency	Severity
MDSE	All	Yes	.99	.08
MDSE	All	No	.98	.18
MDSE	High	Yes	.99	.10
MDSE	High	No	.98	.25
MDSE	Low	Yes	.97	.20
MDSE	Low	No	.97	.18
MSSW	All	Yes	.81	.70
MSSW	All	No	.86	.66
MSSW	High	Yes	.55	.96
MSSW	High	No	.84	.68
MSSW	Low	Yes	.84	.67
MSSW	Low	No	.83	.68

Note that our objective is not for the correlations to necessarily be equal, but that there exists a strong correlation for both frequency and severity. Depending upon the data, the higher frequency CVEs may or may not also be the highest severity CVEs. If so, then the correlations to frequency and severity would both be very high and almost equal. If not, both should still be high but one may be higher than the other. What we do not want in these results is for one of frequency or severity to have a high correlation and the other to have a very low correlation (which can be seen with the MDSE equation).

We also checked to see that the double log still linearized the frequency distribution when using both variants from [11]. While propagating CVEs over the CWEs using the CWE taxonomies and using all applicable CWEs (i.e., pillars, classes, bases, variants, and compounds), the results show that the double log does still linearize the frequency (see Figure 10). The same results were obtained while also performing the experiment using just the pillars/classes and then just the bases, variants, and compounds (graphs not shown).

Using our MSSW equation to aggregate the frequency and severity of CWEs, the top 20 class list for 2019 is shown in Table 2. The top 20 base list is shown in Table 3. These two lists use the modification from [11] where the CVEs are propagated up through the CWE taxonomies. We claim that these two lists represent the most accurate measurement yet produced for determining the most significant software security weaknesses. Given that there is no ground truth for how to best combine frequency and severity and no ground truth upon which to establish the correctness of the CVSS metric, it is likely impossible to prove any such metric as maximally effective. We make our 'most accurate measurement yet' claim based on the demonstrated limitations in the published MDSE equation and a lack of competing published alternatives.

6 DISCUSSION AND ANALYSIS OF THE MOST SIGNIFICANT WEAKNESSES

In this section we evaluate our 2019 MSSW class and base lists (in Tables 2 & 3) and compare them against the 2019 CWE Top 25 MDSE List [16] reproduced in Table 4.

Table 2: 2019 MSSW Top 20 Pillars/Classes, Propagating CVSS Data over CWE Taxonomies

Rank	Identifier	CWE Description	MSSW Score	Frequency	Mean CVSS
1	CWE-913	Improper Control of Dynamically-Managed Code Resources	78.31	188	8.81
2	CWE-119	Improper Restriction of Operations within Bounds of a Memory Buffer	71.14	2745	8.00
3	CWE-669	Incorrect Resource Transfer Between Spheres	64.86	181	8.31
4	CWE-672	Operation on a Resource after Expiration or Release	64.56	876	7.96
5	CWE-330	Use of Insufficiently Random Values	63.74	111	8.43
6	CWE-704	Incorrect Type Conversion or Cast	62.55	54	8.68
7	CWE-287	Improper Authentication	59.75	627	7.86
8	CWE-345	Insufficient Verification of Data Authenticity	54.60	483	7.73
9	CWE-682	Incorrect Calculation	51.94	215	7.78
10	CWE-269	Improper Privilege Management	50.57	258	7.70
11	CWE-610	Externally Controlled Reference to a Resource in Another Sphere	48.38	725	7.46
12	CWE-706	Use of Incorrectly-Resolved Name or Reference	39.04	358	7.23
13	CWE-20	Improper Input Validation	38.56	3960	6.99
14	CWE-116	Improper Encoding or Escaping of Output	32.13	2461	6.82
15	CWE-400	Uncontrolled Resource Consumption	32.07	272	7.01
16	CWE-74	Improper Neutralization of Special Elements in Output ... ('Injection')	32.06	2455	6.82
17	CWE-754	Improper Check for Unusual or Exceptional Conditions	32.05	264	7.01
18	CWE-326	Inadequate Encryption Strength	28.21	35	7.24
19	CWE-668	Exposure of Resource to Wrong Sphere	26.59	2292	6.66
20	CWE-436	Interpretation Conflict	22.40	17	7.19

Table 3: 2019 MSSW Top 20 Bases/Variants/Compounds, Propagating CVSS Data over CWE Taxonomies

Rank	Identifier	CWE Description	MSSW Score	Frequency	Mean CVSS
1	CWE-89	Improper Neutralization of Special Elements used ... ('SQL Injection')	71.70	384	8.89
2	CWE-502	Deserialization of Untrusted Data	61.73	83	9.01
3	CWE-787	Out-of-bounds Write	61.57	423	8.34
4	CWE-78	Improper Neutralization of Special ... ('OS Command Injection')	61.22	194	8.58
5	CWE-120	Buffer Copy without Checking Size of ... ('Classic Buffer Overflow')	59.35	162	8.55
6	CWE-94	Improper Control of Generation of Code ('Code Injection')	58.62	100	8.72
7	CWE-798	Use of Hard-coded Credentials	58.07	89	8.75
8	CWE-434	Unrestricted Upload of File with Dangerous Type	57.95	167	8.46
9	CWE-416	Use After Free	56.69	426	8.09
10	CWE-352	Cross-Site Request Forgery (CSRF)	51.60	386	7.86
11	CWE-346	Origin Validation Error	51.51	430	7.82
12	CWE-613	Insufficient Session Expiration	51.08	402	7.82
13	CWE-190	Integer Overflow or Wraparound	48.79	164	7.95
14	CWE-415	Double Free	43.17	46	8.15
15	CWE-125	Out-of-bounds Read	42.34	658	7.28
16	CWE-129	Improper Validation of Array Index	41.97	25	8.50
17	CWE-611	Improper Restriction of XML External Entity Reference	41.47	100	7.69
18	CWE-918	Server-Side Request Forgery (SSRF)	41.05	74	7.78
19	CWE-22	Improper Limitation of a Pathname to a Restricted ... ('Path Traversal')	39.40	309	7.27
20	CWE-191	Integer Underflow (Wrap or Wraparound)	37.76	18	8.47

As stated previously, we expect the MDSE list to vary from the MSSW class and base lists because

(1) the MDSE list is biased towards the frequency of a CWE occurring in CVEs,
(2) we use the taxonomy propagation approach from [11], and
(3) the class and base lists contain a total of 40 CWEs while the MDSE list contains 25.

6.1 High Level Summaries

View 1003 contains two pillars (CWE-682 and CWE-697) and 36 classes, as well as 81 bases, three variants (CWE-415, CWE-416, and CWE-401), and two compounds (CWE-352 and CWE-384).

The MDSE Top 25 [16] ranks CWE items across all the layers of abstraction from view CWE-1003. The list has seven classes, 16 bases, one variant, and one compound. Of interest is that some of these top CWEs have child-parent relationships among themselves.

Table 4: Reproduction of the 2019 CWE Top 25 Most Dangerous Software Errors List[16]

Rank	Identifier	CWE Description	MDSE Score
1	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
3	CWE-20	Improper Input Validation	43.61
4	CWE-200	Information Exposure	32.12
5	CWE-125	Out-of-bounds Read	26.53
6	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
7	CWE-416	Use After Free	17.94
8	CWE-190	Integer Overflow or Wraparound	17.35
9	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
10	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.1
11	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.47
12	CWE-787	Out-of-bounds Write	11.08
13	CWE-287	Improper Authentication	10.78
14	CWE-476	NULL Pointer Dereference	9.74
15	CWE-732	Incorrect Permission Assignment for Critical Resource	6.33
16	CWE-434	Unrestricted Upload of File with Dangerous Type	5.5
17	CWE-611	Improper Restriction of XML External Entity Reference	5.48
18	CWE-94	Improper Control of Generation of Code ('Code Injection')	5.36
19	CWE-798	Use of Hard-coded Credentials	5.12
20	CWE-400	Uncontrolled Resource Consumption	5.04
21	CWE-772	Missing Release of Resource after Effective Lifetime	5.04
22	CWE-426	Untrusted Search Path	4.4
23	CWE-502	Deserialization of Untrusted Data	4.3
24	CWE-269	Improper Privilege Management	4.23
25	CWE-295	Improper Certificate Validation	4.06

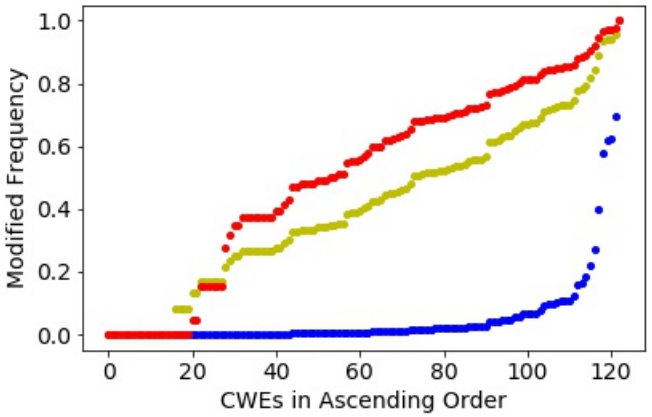

Figure 10: Normalized Distributions of Frequency (bottom blue line), Log of Frequency (middle yellow line), and Double Log of Frequency (top red line).

A simple inspection of the list shows how parent CWEs do not receive CVE counts from their children. For example, the count for the top class CWE-119 (rank 1, count 1048) does not include the counts of its children CWE-125 (rank 5, count 678) and CWE-787 (rank 12, count. 473). Analogously, the count for the class CWE-287 (rank 13, count 299) does not include the counts of its children base CWE-798 (rank 19, count 91) and base CWE-295 (rank 25, count 77).

Our MSSW class list is comprised of 19 class CWEs and the pillar CWE-682 (rank 9) – see Table 2. Only three CVEs are directly described with the pillar but it appears in the list because there is a set of severe CVEs described with its children (see subsection 6.5). Our MSSW base list is comprised of 17 bases, the variants CWE-416 (rank 7) and CWE-415 (rank 14), and the compound CWE-352 (rank 10) –see Table 3. Each of the two lists properly compare items of the same kind. Interestingly but not surprisingly, each CWE from the base list is a child of a CWE from class list. However, the ordering of these parent-child pairs are not necessarily preserved between the two lists.

6.2 Set Differences

There are differences in the set of CWEs covered by our top 20 MSSW class and base lists and the MDSE list. The pillars/classes in the MDSE list that do not appear in the class list are: CWE-200 and CWE-732. The bases/variants/compounds in the MDSE list that do not appear in the base list are: CWE-79, CWE-476, CWE-772, CWE-426, CWE-295. The base list contains base CWE-120 (a child of CWE-119) but this does not appear in the MDSE list.

Note that the two classes from the MDSE list with children in the same list are also in the class list (emphasizing their importance): class CWE-119 with children base CWE-125 and base CWE-787, and class CWE-287 with children base CWE-798 and base CWE-295.

6.3 Reordered Rankings

The relative orderings in the MDSE list often do not match the orderings in the MSSW class and base lists. There are some notable reorderings. CWE-89 (Structured Query Language (SQL) Injection) and CWE-502 (Deserialization of Untrusted Data) climb up in the base list due to their highest severities of 8.89 and 9.01. CWE-913 (Improper Control of Dynamically-Managed Code Resources) does not even appear in the MDSE Top 25 list, as it has only three direct occurrences in CVE. However, it climbs up to first position in the class list due to its highest severity of 8.81 and its 188 propagated occurrences. Its main child contributor is base CWE-502 with frequency of 83 and severity of 9.01. CWE-119 (Improper Restriction of Operations within the Bounds of a Memory Buffer) in the MDSE list, while widely used with 2745 propagated occurrences in the CVEs, is quite less severe than CWE-913 and drops to rank 2 in the MSSW class list.

6.4 The Two Most Dangerous CWEs: Injection vs. Memory Errors

The two most distinctive groups of weaknesses both in the MDSE Top 25 list and the two MSSW Top 20 lists are injection and memory errors. However, the use of the MSSW equation and the split into the class and base lists considerably reorders these two groups, as well as bringing in new CWEs and dropping some CWEs pertaining to those groups.

6.4.1 Injection Weaknesses. Injection is the most dangerous type of weakness, represented by CWE-89 (SQL Injection), CWE-78 (OS Command Injection), CWE-94 (Code Injection), and CWE-611 (Improper Restriction of Extensible Markup Language (XML) External Entity Reference), with ranks 1, 4, 6, and 17 respectively in the base list (see Table 3). An injection happens when a command string gets assembled partially from input with language-specific special elements and parsed into an invalid construct [3]. The invalid constructs for the four most dangerous injection CWEs are respectively SQL query, OS command, code, XML entity. The MDSE list also contains these four CWEs, however the rankings of the first two are 6 and 11 due to their lower frequencies of 397 and 217. The MSSW inclusion of their high severity scores of 8.86 and 8.56 moved them several positions up in the base list.

Also of importance is that the second ranked in the MDSE list CWE-79 (Cross-site Scripting), where the invalid construct is a generated Web Page, is not in our MSSW base list. Although it has the highest frequency of 1571, its severity score of 5.83 is relatively low.

The MSSW class list includes CWE-116 (Improper Encoding or Escaping of Output) and CWE-74 (Injection), ranked 14 and 16 (see Table 2). The reason for that is CWE-116 is a typical cause of injection and CWE-74 is the parent of CWE-78, CWE-89, and CWE-94. Interestingly, the class CWE-74 has rank 16 among classes, while its children base CWE-89, CWE-78, and CWE-94 are ranked 1, 4, 6 among bases. The frequencies of 2455 for CWE-74, 384 for CWE-89, 194 for CWE-78, and 100 for CWE-94, leave 1777 injection CVEs that are described with CWEs that are either very infrequent or not severe. These are CWE-79 (Cross-site Scripting) with the low severity of 5.83, CWE-88 (Argument Injection) with the low frequency of 6, and CWE-91 (XML Injection) with the low frequency of 16. Being not too dangerous they bring the class CWE-74 down to rank 16. That same base CWE-79, not included in the MSSW base list, is ranked 2nd in the MDSE list due to the frequency biasing.

6.4.2 Memory Weaknesses. The most dangerous memory weaknesses are CWE-787 (Out-of-bounds Write) and CWE-120 (Classic Buffer Overflow) with ranks 3 and 5 – see Table 3. Both of them are included in the base list but not the MDSE list, due to the correction of the frequency bias towards proper inclusion of their severity scores of 8.34 and 8.55.

The other memory weaknesses in the MSSW class and base lists are as follows:

- bases CWE-120, CWE-125, and CWE-787 are buffer overflow (out of bounds read or write)
- variant CWE-416 is use after free (use of deallocated memory through a dangling pointer)
- variant CWE-415 is double free (deallocate of already deallocated memory)
- class CWE-119 is a general memory corruption weakness, which includes buffer overflow, use after free and double free.
- class CWE-400 is memory overflow (stack/heap exhaustion) [20]

6.4.3 Injection/Memory Weakness Comparison. Compared to MDSE, the MSSW equation brings up several injection weaknesses with much higher severity than that of any memory weaknesses. The related CVE analysis confirms that the injection CVE are easier to exploit and with higher impact. An injection directly leads to arbitrary command, code, or script execution. Once a SQL injection is in place, there is no need of additional sophisticated attack crafting or use of glitches in the system. However, it takes considerable extra effort for an attacker to turn a buffer overflow into an arbitrary code execution. The attacker also needs exceptional skills in applying spraying memory techniques. The possible damage from an SQL injection is also very high. It may expose huge amounts of structured data, which is always more valuable than raw data. Well formed structured data is easy to read, sort, search, and make sense of it. An attacker could modify a database – insert, update, delete data, execute admin operations, recover file content, and even issue OS commands [24].

6.5 Next Most Dangerous CWEs

The next most dangerous groups of weaknesses in the MSSW class and base lists relate to file input and upload, authentication, randomization, cryptography, arithmetics and conversion, and input validation:

- file input and upload - base CWE-502 (Deserialization of Untrusted Data) and base CWE-434 (Unrestricted Upload of File with Dangerous Type) have ranks 2 and 8 respectively. They are the main contributors to class CWE-913 and class CWE-669 with ranks 1 and 3, respectively.
- authentication - base CWE-798 (Use of Hard-coded Credentials) has rank 7; it is one of the contributors to the class CWE-287 (Improper Authentication) with the same rank 7 in the class list.

- randomization - class CWE-330 (Use of Insufficiently Random Values) with rank 5 is the class mostly directly assigned to CVEs.
- cryptography - base CWE-352 (Cross-Site Request Forgery) has rank 10, which relates to bugs in data verification. The class list also has class CWE-326 (Inadequate Encryption Strength) with rank 18, which is directly assigned to 35 CVEs with severity 7.24.
- arithmetics and conversion - base CWE-190 (Integer Overflow or Wraparound) and base CWE-191 (Integer Underflow) have ranks 13 and 20. They are the primary contributors to pillar CWE-682 (Incorrect Calculation) with rank 9. Others in this group on the top lists are bases CWE-131 (Incorrect Calculation of Buffer Size), CWE-190 (Integer Overflow or Wraparound), and CWE-191 (Integer Underflow – Wrap or Wraparound).
- input validation - base CWE-129 (Improper Validation of Array Index) has rank 16.

6.6 Mapping Dependencies

Both the MDSE and MSSW rankings heavily depend on how NVD assigns CWEs to particular CVEs. The CWE selection is restricted to view CWE-1003. Insufficient information about a CVE or an insufficiently specific CWE may lead to the use of the closest matching CWE class or pillar to describe the CVE. For example, it makes sense for class CWE-119 to be used for the memory corruption CVE-2019-7098, as there is not much information (no code and no details) – it could be any memory use error or a double free. However, there does exist enough information about the use after free CVE-2019-15554 but it is still mapped to class CWE-119 because there exists no appropriate base CWE. A close base CWE is CWE-416 (Use After Free), but it does not really reflect memory safe languages like Rust. It is also possible for a class CWE to be assigned to a CVE even when a specific base CWE is available. For example, the stack buffer overflow write CVE-2019-14363 is assigned class CWE-119, although there is plenty of information to map more specifically to bases CWE-121 and CWE-120.

7 RELATED WORK

The constant need to improve information security has motivated a widespread interest in metrics (both qualitative [7] and quantitative [22]). As stated by Lord Kelvin, *you cannot improve if you cannot measure*. However, many members of the software security community doubt our ability to quantify security. Bellovin was among the first [2] to argue about the infeasibility of software security metrics. [4] discusses the limitations of the celebrated "Risk = Threat × Vulnerability × Consequence" model that is widely used. In [28] Verendel presents a critical survey of results and assumptions made in the community to quantify security. After reviewing over 100 articles, he concludes that the validity of most methods is still strikingly unclear. Many reasons explain this invalidity: lack of validation, lack of comparison against empirical data, and the fact that many assumptions in formal treatments are not empirically well-supported in operational security.

Although we agree, we posit that acceptable but possibly imperfect metrics must be developed in order to facilitate security decisions and to evaluate changes in security posture. To this end, there have been substantial efforts to produce security metrics; [28] surveys the literature of security metrics published between 1981 and 2008. More efforts can be found in [26], [25], and [19]. Security metrics that produce lists of the top security issues are also very prevalent [27], [10]. Specific to software security, there is the OWASP Top 10 [23] for web applications. Also, the CWE project has the Common Weaknesses Scoring System (CWSS) [13] and the Common Weakness Risk Analysis Framework (CWRAF) [15], which are used together to provide the most important weaknesses tailored to a particular organization.

There is also work to critique and improve the foundational data structures used by the MDSE and MSSW metrics. CWEs have been discussed in [29]. An entirely new approach to classifying software bugs (weaknesses) is proposed by [3] and is currently under development. The evolution of CWE is documented in [18] (e.g., the addition of classification trees and content for mobile applications and hardware). A critique of CVSS is available in [8]. In [11] a novel CWE data collection method is proposed along with simple atomic software security metrics. Our approach in contrast is an aggregate metric designed to be a direct replacement for the MDSE equation.

Along with much other work, our research should be considered as an important step in the process to improve CWE. We believe that our contribution is major as it points out a serious bias in the CWE MDSE equation that is preventing accurate measurements of the most significant software security weaknesses.

8 CONCLUSION

The field of security metrics is a difficult area of scientific research because there is often no ground truth, unlike disciplines such as physics and chemistry. This may lead one to focus on just taking simple low level measurements that are inherently defensible; that was the approach taken in [11]. However, creating aggregate metrics that compose multiple simple measurements is of practical importance for the field of security. In this work we did just that, aggregating frequency and severity (i.e., exploitability and impact) into a single metric. Our objective is not for the correlations to necessarily be equal, but that there exists a strong correlation for both factors which more evenly balances the inclusion of the top frequency and top severity CWEs. This seemingly simple task proved challenging because of the differing distributions of both simpler metrics. Indeed, the officially published CWE metric neglected this property and did not achieve its stated objective (almost exclusively choosing the most frequent CWEs). With our work, we claim to have addressed the limitations and to have produced the most accurate equation yet for measuring the most significant software security weakness.

REFERENCES

[1] David W Baker, Steven M Christey, William H Hill, and David E Mann. 1999. The Development of a Common Enumeration of Vulnerabilities and Exposures. In *Recent Advances in Intrusion Detection*, Vol. 7. Online proceeding, Purdue, IN, USA, 9.

[2] Steven M. Bellovin. 2006. On the Brittleness of Software and the Infeasibility of Security Metrics. *IEEE Security and Privacy* 4, 4 (July 2006), 96. https://doi.org/10.1109/MSP.2006.101

[3] Irena Bojanova, Paul E Black, Yaacov Yesha, and Yan Wu. 2016. The Bugs Framework (BF): A Structured approach to express bugs. In *2016 IEEE International*

Conference on Software Quality, Reliability and Security (QRS). IEEE, IEEE Press, Vienna, Austria, 175–182.

[4] Louis Anthony (Tony) Cox, Jr. 2008. Some Limitations of "Risk = Threat × Vulnerability × Consequence" for Risk Analysis of Terrorist Attacks. *Risk Analysis* 28, 6 (2008), 1749–1761. https://doi.org/10.1111/j.1539-6924.2008.01142.x

[5] FIRST. 2019. Common Vulnerability Scoring System Special Interest Group. https://www.first.org/cvss Accessed: 2019-12-10.

[6] FIRST. 2019. Common Vulnerability Scoring System v3.1: Specification Document. https://www.first.org/cvss/v3.1/specification-document Accessed: 2020-2-5.

[7] Debra S. Herrmann. 2007. *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI* (1st ed.). Auerbach Publications, USA.

[8] Allen D. Householder Art Manion Deana Shick Jonathan Spring, Eric Hatleback. 2018. Towards Improving CVSS. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=538368 Accessed: 2020-05-11.

[9] Robert A. Martin and Sean Barnum. 2008. Common Weakness Enumeration (CWE) Status Update. *Ada Lett.* XXVIII, 1 (April 2008), 88–91. https://doi.org/10.1145/1387830.1387835

[10] McAfee. 2020. McAfee Labs 2019 Threats Predictions Report. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-labs-2019-threats-predictions/ Accessed: 2020-02-01.

[11] Peter Mell and Assane Gueye. 2020. A Suite of Metrics for Calculating the Most Significant Security Relevant Software Flaw Types. In *2020 Conference on Computers, Software and Applications (COMPSAC)*. IEEE, IEEE Computer Society Press, Madrid, Spain.

[12] MITRE. 1999. Common Vulnerabilities and Exposures. https://cve.mitre.org Accessed: 2020-2-5.

[13] MITRE. 2018. Common Weakness Scoring System (CWSS). https://cwe.mitre.org/cwss/ Accessed: 2020-04-10.

[14] MITRE. 2019. Common Weakness Enumeration. https://cwe.mitre.org Accessed: 2019-12-10.

[15] MITRE. 2019. Common Weakness Risk Analysis Framework (CWRAF). https://cwe.mitre.org/cwraf/ Accessed: 2020-04-10.

[16] MITRE. 2020. 2019 CWE Top 25 Most Dangerous Software Errors. https://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html Accessed: 2020-02-01.

[17] MITRE. 2020. CWE Glossary. https://cwe.mitre.org/documents/glossary/ Accessed: 2020-05-11.

[18] MITRE. 2020. History of the Common Weakness Scoring System (CWSS). https://cwe.mitre.org/about/history.html Accessed: 2020-04-10.

[19] Patrick Morrison, David Moye, Rahul Pandita, and Laurie Williams. 2018. Mapping the field of software life cycle security metrics. *Information and Software Technology* 102 (2018), 146 – 159. https://doi.org/10.1016/j.infsof.2018.05.011

[20] NIST. 2020. BF Memory Model. https://samate.nist.gov/BF/Classes/MEMModel.html Accessed: 2020-05-11.

[21] NVD. 2020. National Vulnerability Database. https://https://nvd.nist.gov Accessed: 2020-01-10.

[22] Xinming Ou and Anoop Singhal. 2011. *Quantitative security risk assessment of enterprise networks*. Springer-Verlag, New York, NY, USA.

[23] OWASP. 2020. OWASP Top Ten. https://owasp.org/www-project-top-ten/ Accessed: 2020-04-10.

[24] OWASP. 2020. SQL Injection. https://owasp.org/www-community/attacks/SQL_Injection Accessed: 2020-05-11.

[25] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2016. A Survey on Systems Security Metrics. *ACM Comput. Surv.* 49, 4, Article 62 (dec 2016), 35 pages. https://doi.org/10.1145/3005714

[26] T. W. Purboyo, B. Rahardjo, and Kuspriyanto. 2011. Security metrics: A brief survey. In *2011 2nd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*. IEEE, Bandung, Indonesia, 79–82.

[27] Symantec. 2020. 2019 Internet Security Threat Report. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf Accessed: 2020-02-01.

[28] Vilhelm Verendel. 2009. Quantified Security is a Weak Hypothesis: A Critical Survey of Results and Assumptions. In *Proceedings of the 2009 Workshop on New Security Paradigms Workshop (NSPW '09)*. Association for Computing Machinery, New York, NY, USA, 37–50. https://doi.org/10.1145/1719030.1719036

[29] Y. Wu, Irena Bojanova, and Y. Yesha. 2015. They know your weaknesses - Do you?: Reintroducing Common Weakness Enumeration. *CrossTalk* 28 (01 2015), 44–50.