# Improved Prediction of Vulnerability Exploitation using CVSS Base-Score with Optimized Equation Parameters

Ben Church

**Abstract—Introduction:** Vulnerability risk assessment is important for organizational software reliability maintenance. Vulnerabilities must be prioritized so companies can maximize the utility of their limited resources available for redressment. The CVSS base score was designed to provide a standardized, quantified measure of the risk presented by a vulnerability, however, it correlates poorly with real vulnerability exploitation. The numerical values used for different vulnerability metrics cases in the base score equation have arbitrary values. We therefore optimized these values for CVSS base score exploit prediction. **Related Work:** An overview is given of other works done to improve CVSS risk remediation or exploit prediction. Existing works done to reduce the cost of vulnerability prioritization by considering the context-specific CVSS scores depend on the costs associated with the context-particular vulnerability addressment strategy. The works done to extend the CVSS base score with machine learning on natural language features of vulnerability descriptions produce similar context-specific results, leaving the issue of standardization unaddressed. Works which evaluated the CVSS, or other prediction systems, are also explored. They introduce useful prediction performance metrics which were used to drive the optimization and validate the resulting prediction systems. **Methods:** A random sample of 10,000 entries form the National Vulnerability Database (NVD), from between the years 2010 and 2016, was used for two optimization approaches. One approach combined the system's sensitivity and precision into an objective function, and the other used risk reduction as the objective function. Those metrics, and the intra-class correlation coefficient, were used to validate the optimized prediction systems' performances on 5 new samples of 1000 NVD entries. **Results:** Neither approach resulted in significant risk reduction improvements. Prediction sensitivity improved using both optimization approaches, but always with the cost of over estimating risk and generating additional false positives. 'Exploited' and 'unexploited' prediction classes gained no additional distinctness from one another, and other performance metrics, including precision, and intra-class correlation coefficients exhibited no substantial improvements. **Discussion & Conclusions:** The results indicate limited improvements in exploit prediction. This may because, unlike sensitivity, optimizing risk reduction and precision require balancing opposing objectives, risk must be identified without false positives. Apparently, this balance cannot be achieved with the existing vulnerability metrics. Future work might investigate the use of new metrics indicating risk reductions entailed by vulnerability characteristics to balance sensitivity with precision. Once such metrics are chosen, the methods presented herein could be used to obtain optimal scoring equation values.

**Index Terms**—Vulnerability, CVSS, base score, exploit, prediction, optimized

## 1 INTRODUCTION

Software vulnerability addressment is an important problem for organizations since exploitation of such vulnerabilities causes preventable losses. This problem tends to grow with time because vulnerabilities are often reported more frequently than they can be fixed. To minimize losses in spite of the increasing number of vulnerabilities becoming known, and with limited available resources, organizations must prioritize their vulnerability addressment strategy. Organizations minimize real losses by prioritizing vulnerabilities based on the cost each one is expected to impart if left unaddressed. Probabilistically, this expected cost, commonly called risk, is equal to the real cost in the event of exploitation, times the probability of that exploitation occurring.

The Common Vulnerability Scoring System (CVSS) was created to provide a single, objective measure of the risk presented by any software vulnerability [1]. The CVSS offers three scores in the range of 0-10, measuring the risk presented by the vulnerability. They are the base, temporal, and environmental scores. The base score coveys the intrinsic vulnerability risk, or time and context invariant risk, by considering six mostly objective vulnerability metrics. The temporal and environmental scores essentially provide context specific re-evaluations of the base score. The temporal score takes into consideration changes in public or expert vulnerability knowledge, or vulnerability patch deployment. Through the temporal score, one might achieve a more accurate measure of the risk currently posed by a vulnerability. The environmental score considers the distribution of the vulnerability in the computer system or organization of interest, as well as the subjective importances of various computer system services which could be compromised. Because of their (mostly) objective and unchanging nature, CVSS base scores are widely available from vulnerability databases and are often used in research.

To reflect the risk conveyed with a vulnerability, the CVSS base score equation uses an impact, and an exploitability metric group, each with 3 metrics. The impact term in the equation reflects the severity of the consequences of the vulnerability being exploited. The exploitability term

• *B. Church is with the School of Computing, Queen's University, Kingston, ON, Canada. E-mail: ben.church@queensu.ca*

is meant to reflect the probability that the vulnerability will be exploited by using metrics which reflect the difficulty of exploitation. The complete formula for the CVSS base score equation v2 is shown in Figure 1.

BaseScore = 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

AccessVector = case AccessVector of
requires local access: 0.395
adjacent network accessible: 0.646
network accessible: 1.0

AccessComplexity = case AccessComplexity of
high: 0.35
medium: 0.61
low: 0.71

Authentication = case Authentication of
requires multiple instances of authentication: 0.45
requires single instance of authentication: 0.56
requires no authentication: 0.704

ConfImpact = case ConfidentialityImpact of
none: 0.0
partial: 0.275
complete: 0.660

IntegImpact = case IntegrityImpact of
none: 0.0
partial: 0.275
complete: 0.660

AvailImpact = case AvailabilityImpact of
none: 0.0
partial: 0.275
complete: 0.660

*Figure 1: CVSS base score equation, v2 [1]*

To the end of providing an objective scoring system, most of the metrics in the equation are objective, and where possible, quantitative. Variation in a particular vulnerabilities scoring cannot come from user subjectivity in whether zero, one, or multiple instances of user authentication are required for exploitation. Likewise, whether a system's data confidentiality is not at all compromised, partially compromised, or totally compromised, is objective and quantitative. A notable exception to the objective metrics is the AccessComplexity metric, which can be high, medium, or low. The CVSS has in fact received criticisms of subjectivity [2, 3].

Despite attempts to provide the CVSS as a vulnerability risk scoring system with metrics both objective enough to provide consistent vulnerability scoring, and comprehensive enough to accurately assess the vulnerability, corporations often resort to their own assessment systems. Younis and Malaiya [4] performed a study on the effectiveness of the CVSS base score and Microsoft's vulnerability rating systems for predicting vulnerability exploitation. They correlated predictions based on CVSS base scores and Microsoft system scores with real exploitation as determined by the existence of Exploit Database[1] (EDB) entries. Both systems performed similarly, in that they were both sensitive enough to detect most exploitable vulnerabilities, but they had false positive rates, above 90% for vulnerabilities found in Internet Explorer.

Allodi and Massacci [5] performed a case-control study in which they examined the effect which fixing vulnerabilities based on their CVSS base scores had on CVSS exploit prediction. Vulnerabilities were considered exploited if they had an entry in a number of databases, including the EDB. Therefore they were able to measure the CVSS' exploit prediction capabilities in terms of how many vulnerabilities with scores above a threshold had EDB references. They concluded that the CVSS is no better than guessing randomly for exploit prediction.

The failure of the CVSS to reliably predict exploitation in the wild is illustrated both in literature such as [4, 5], and by the adoption of organization-specific vulnerability rewards programs (VRPs) such as Mozilla's[2] or Google's[3]. Nonetheless, the system's open availability, the amount of published research, and the official recognition from organizations such as the National Institute of Standards and Technology under the Security Content Automation Protocol[4] (SCAP) make the CVSS an attractive starting point for the development of a better scoring system.

The aim of this work was to improve the ability of the CVSS base score to predict vulnerability exploitation by optimizing parameters in the base score equation. Despite efforts made to use objective and quantitative metrics, and regardless of whether some may be subjective, the numerical values for the metrics cases are apparently arbitrary. The values exhibit a natural order in the sense that they increase as the metric cases' riskiness increase, however, there is no reason that all the Impact metrics' cases should have the same values, for instance. These 18 equation parameters were determined by optimizing CVSS prediction performance metrics, defined in the Related Work section below. The optimized versions of the equation were assessed for their ability to predict exploits in new samples of vulnerabilities to investigate the feasibility of obtaining accurate exploit prediction with the CVSS.

The remainder of this paper is organized as follows: In the Related Work section, works are explored which investigated improving or extending the CVSS base score. Several methods and metrics for scoring system performance measurement are also introduced. The Methods section first introduces the data used for the experiment, then describes the optimization process, and finally reports the metrics and methods used to assess the optimized CVSS' performance. Vulnerability base score distributions, exploit prediction results, and performance metrics are reported in the Results section. Interpretations, implications, and limitations of the results are explored in the Discussion & Conclusions section with considerations for future improvements.

[1] https://www.exploit-db.com

[2] https://www.mozilla.org/en-US/security/bug-bounty/

[3] https://www.google.com/about/appsecurity/reward-program/

[4] https://scap.nist.gov/index.html

## 2 RELATED WORK

Frühwirth and Männistö [6] investigated the benefits to vulnerability prioritization when the additional Temporal and Environmental metric scores are used. They assigned relative risk group values to vulnerabilities depending on the location of its base score on the interval 0-10. They then assigned cost scores to vulnerabilities of the different risk groups, reflecting the losses a company would incur by the method of redressment chosen given the severity of the risk. This allowed a comparison between the expected cost for a given set of vulnerabilities when only the base score is used for prioritization, and the expected cost when the Temporal and Environmental score were used as well. Although their results indicate cost reductions, this is because of higher severity vulnerabilities, with more expensive redressment procedures, being re-evaluated as lower severity. They assume that all vulnerabilities are eventually fixed, and only take into account the cost of fixing the vulnerability, not the cost of having it exploited and corresponding risk. Their cost reduction results, if only for prioritization costs and not risk, reflect the utility which additional information may have for exploit prediction.

Tripathi and Sighn [7] investigated the ability for adjusted vulnerability severity scores to rank vulnerability classes. The severity scores were based on the CVSS scores, and an age-decay factor if the vulnerability was over a month old, corresponding to the existence of a patch in their framework. The vulnerability classes are simple descriptions of the mechanics of the attack corresponding to the vulnerability, such as 'buffer overflow' or 'SQL injection'. These classes are provided by the NVD. The 10 most severe vulnerability types found by their study had some correspondence with several security advisories vulnerability class lists, but this correspondence was not quantified. Furthermore, their method depends on several assumptions, such as invariable decrease in vulnerability severity with time, while the reality may run counter to this as attackers learn of the vulnerability. Therefore, their methods are not considered for use in the context of standardization in this paper, for the difficulty in justifying the assumptions required to compute their metrics.

Bozorgi *et al.* [3] and Khazaei *et al.* [2] proposed machine learning methods for automated base score prediction using additional information from vulnerability database entries. They proposed these methods in response to questions of the ability of the CVSS base score to predict actual exploitation of vulnerabilities. While machine learning may be useful for correlating vulnerability information with base scores, or even predicting exploitation, it probably cannot be used to achieve the single scoring system intended by the CVSS. If machine learning was integrated into the CVSS, different users of the system could not take advantage of online learning from new vulnerabilities. System specific online learning will cause the system's model to change, potentially scoring a vulnerability differently than another system. Otherwise, it cannot improve itself from new information. As such, the need remains for a single, standardized vulnerability scoring system.

To evaluate the CVSS' exploit prediction capabilities, both Allodi and Massacci [5], and Younis *et al.* [8] calculated the system's confusion matrix. A confusion matrix simply shows the number of vulnerabilities (or predictions, generally) which fall into each of four possible groups. A general confusion matrix is shown in Table 1.

*Table 1: Prediction system confusion matrix structure*

| | | Actually exploited? | |
|---|---|---|---|
| | | True | False |
| Exploit predicted? | True | True positive (TP) | False positive (FP) |
| | False | False negative (FN) | True negative (TN) |

The elements of the confusion matrices are then used to compute a number of more abstract performance measures. Allodi and Massacci [5] measured prediction system performance in terms of risk reduction (RR). They formulate RR as the probability of a vulnerability, V, having an EDB entry, $E_V$, given that its base score was above a threshold, T, minus the probability that the vulnerability did not appear in EDB given that its base score was below the threshold. This is shown probabilistically in equation 1, and in terms of confusion matrix elements in equation 2.

$$RR = P(E_V|score(V) > T) - P(E_V|score(V) < T) \quad (1)$$

$$RR = \frac{TP}{(TP+FP)} - \frac{FN}{(FP+TN)} \quad (2)$$

To compare the effectiveness of the CVSS and corporation specific VRPs, Younis *et al.* [8] computed sensitivity, precision, and a weighted average combination of them. Definitions for the sensitivity and precision are given in equations 3, and 4.

$$Sensitivity = \frac{TP}{TP+FN} \quad (3)$$

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

Thus, sensitivity is the ratio of exploits successfully predicted to the number of exploits which should have been predicted. Precision is the system's ability to predict without over-predicting.

Despite the works done on extending the base score with contextual information, or predicting it with natural language and machine learning, this is the first work which

seeks to improve exploit prediction by only altering scalar values in the base score equation, to the best of my knowledge. The works done to assess the performance of the CVSS for exploit prediction provide many of the metrics used for both optimization and validation.

## 3 METHODS

An overview of the experimental procedure is shown in Figure 2, while details of the procedure are provided throughout the rest of the section.

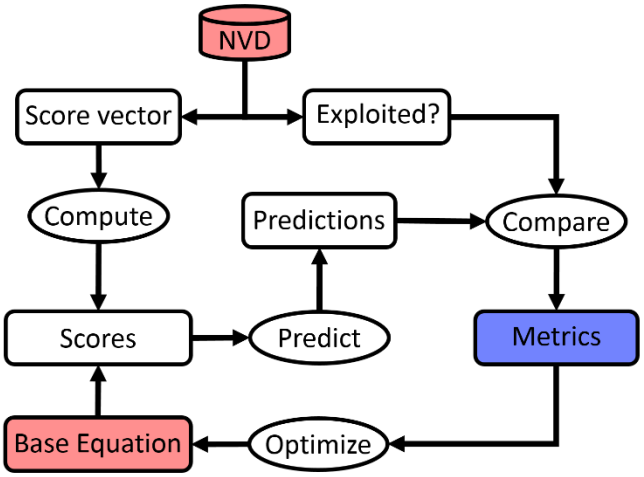


*Figure 2: Conceptual overview flowchart of experiment*

Red shapes indicate openly available data and information from which the experiment begins. Blue shapes represent output from the workflow which can be used to measure system performance. They are color coded in this way because both the base equation parameters, and its resulting performance metrics form a feedback loop during optimization; the base equation is only initial information before its parameters are modified through optimization, and the metrics only reflect system performance once optimization is finished.

The red cylindrical block is the National Vulnerability Database[5] (NVD) from which vulnerability base score vector metrics and EDB references are retrieved. The base equation is used to assess vulnerabilities from their metric cases, which are then compared to whether the NVD entry has an EDB reference. The resulting prediction performance metrics are used to define objective functions, which drive the optimization, modifying the base score equation for the next iteration. The performance metrics after optimization are the measure of the effectiveness of the prediction system. These experiment components are discussed in greater detail in their respective subsections.

### 3.1 Data

To optimize the CVSS base score equation for exploit prediction, both vulnerability exploit existence and base score data were required. The NVD was used as the source of this data. The NVD was chosen for the prospect of a large, unbiased set of vulnerability entries, by virtue of its 80,000 plus entries and governmental operation. The vulnerability entries from between the years 2010 and 2016 (inclusive) were chosen for this study. This was to avoid including increasingly dated and therefore irrelevant software and vulnerabilities, while allowing a chance for vulnerabilities to be reported by not including everything up to the present.

10,000 entries were randomly selected from this range and used to optimize the equation parameters. After optimization, 5 sets of 1000 vulnerabilities were selected and used to assess the performance of the CVSS after optimization. The six element base score vectors, containing the vulnerability's metric cases, rather than the final numerical score, were extracted from the database entries so base scores, and the resulting classification performance metrics could be recomputed to drive the optimization.

Vulnerabilities were classified into 'exploited' or 'unexploited' based on whether the NVD entry had a reference to the EDB. EDB references were also used by Allodi and Massacci [5], and Younis *et al.* [8] as a ground-truth, exploit-existence classification characteristic. It is used in this study because of its large size (over 36,000 entries) compared to other exploit databases, such as Contagio's[6] exploit kits, or Symantec's AttackSignature[7] (roughly 6000 entries) or ThreatExplorer[8] (1172 entries) public datasets used by Allodi and Massacci [5] in addition to the EDB. The EDB may provide a less biased and more extensive sampling of vulnerability exploits than a corporation-specific database, run for their own interests.

### 3.2 Optimization

Risk reduction, sensitivity, and precision, as they are formulated above, cannot be used for an optimization objective function. Small changes in base score equation parameters occurring during optimization result in no change in the metrics. This is because the classification system is discrete; small changes in the parameters will not usually push a vulnerability's base score past the classification threshold. Therefore they lack a gradient required for the optimization. To remedy this, total over-prediction error (TOE) and total under-prediction error (TUE) were introduced as shown in equations 5 and 6.

$$TOE = \sum_{i=0}^{unexploited} \begin{cases} score_i - T \; if \; score_i > T \\ 0 \; otherwise \end{cases} \quad (5)$$

$$TUE = \sum_{i=0}^{exploited} \begin{cases} T - \; score_i \; if \; score_i < T \\ 0 \; otherwise \end{cases} \quad (6)$$

[5] https://nvd.nist.gov/

[6] http://contagiodump.blogspot.ca/2010/06/overview-of-exploit-packs-update.html

[7] https://www.symantec.com/security_response/attacksignatures/

[8] https://www.symantec.com/security_response/landing/azlisting.jsp

where T is the score threshold for classification. TOE and TUE provide a continuously variable measures of a system's over and under prediction errors, respectively. For optimization, they were averaged and normalized over the range 0-1 into the unit over-prediction error (UOE) and unit under-prediction error (UUE) shown in equations 7 and 8.

$$UOE = \frac{TOE}{num_{unexploited} \times (10 - T)} \quad (7)$$

$$UUE = \frac{TUE}{num_{exploited} \times T} \quad (8)$$

$num_x$ is the number of members in class x. The first of two objective functions, $Obj_1$, was computed from RR and UUE as follows:

$$Obj_1 = \frac{((1 - RR) + (UUE))}{2} \quad (9)$$

To the end of using sensitivity and precision for optimization as well, these continuous, normalized measures of classification error were combined with sensitivity and precision into two F-measures, $F_1$ and $F_2$, shown in equations 10 and 11.

$$F_1 = \frac{(1 - Precision) + UOE}{2} \quad (10)$$

$$F_2 = \frac{(1 - Sensitivity) + UUE}{2} \quad (11)$$

$F_1$ and $F_2$ were defined this way so better system performance is indicated by values closer to 0, since the optimization algorithm used minimization. The F values were then combined in a weighted average into an F-measure for the second optimization objective function value, $Obj_2$, shown in equation 12. The average was weighted this way to balance an observed tendency for sensitivity to improve with optimization, but not precision.

$$Obj_2 = \frac{3 \times F_1 + F_2}{4} \quad (12)$$

10,000 vulnerabilities were selected from the 2010-2016 set, and a set of base score equation parameters was generated by optimizing using each objective function. The optimization was performed using Scipy's[9] optimization toolbox. The sequential least squares programming method was used because supported bounding the parameters values between 0 and 1. The equation forms obtained by optimizing its parameters for both objective functions were then validated by generating prediction results for new samples of vulnerabilities.

### 3.3 Classification assessment

To validate that optimization had not caused the system to over-learn the data set, and that it could accurately predict exploitation of new vulnerabilities, 5 additional samples of 1000 vulnerabilities were selected. Their base scores were computed using the new equation forms, and normalized over the range 0-10. These base score were compared to a threshold of 7 which marked the transition to 'high exploitability' used by Younis and Malaiya [4]. Normalization required for comparison because the original equation is designed to return only scores in this range. Scores were normalized simply by subtracting the most negative score, if there was a negative score, in the set, and multiplying by the ratio of 10 over the largest score, as described in equation 13.

$$\forall s_i \in Scores, \bar{s_i} = \begin{cases} (s_i - \min(Scores)) \times \left(\frac{10}{\max(Scores)}\right) & if\ \min(scores) < 0 \\ s_i \times \left(\frac{10}{\max(Scores)}\right) & otherwise \end{cases} \quad (13)$$

Risk reduction, sensitivity, and precision, as defined above, were computed as measures of exploit prediction performance. Additionally, an intra-class correlation (ICC) coefficient was defined to provide an indirect measure of system performance. This definition is shown in equation 15:

$$ICC_x = \frac{var_{x-A}}{var_{x-A} + vat_{x-x}} \quad (14)$$

The subscript x denotes the class within population A. $var_{x\text{-}A}$ is the variability class x experiences relative population A's mean value, that is, the distinctness of the class from the general population. $var_{x\text{-}x}$ is the variability within class x, that is, a reflection of how unlikely it is that a datum like those of class x belongs to class x. The ICC therefore provides a measure of how confident we are in a classification. The ICC was not used for optimization because it was found to 'short-circuit' the classification system; $var_{x\text{-}x}$ was driven to nearly 0, causing many false positive results, and preventing distinction between vulnerabilities with either class.

## 4 Results

Table 2 shows the equation parameters from before and after optimization by both methods. They do not reflect the exploit prediction performance or optimization performance directly, but can be used to infer information about the utility of the various metrics.

[9] https://www.scipy.org/

*Table 2: CVSS base score equation parameters before, and after both optimization approaches*

| | | Optimization method | | |
|---|---|---|---|---|
| **Metric** | **Case** | **None** | **RR** | **F-measure** |
| Access Vector | L | 0.395 | 0.624 | 0.384 |
| | A | 0.646 | 0.646 | 0.642 |
| | N | 1.000 | 0.972 | 0.918 |
| Access Complexity | H | 0.350 | 0.510 | 0.353 |
| | M | 0.610 | 0.782 | 0.707 |
| | L | 0.710 | 0.942 | 0.589 |
| Authentica-tion | M | 0.450 | 0.450 | 0.450 |
| | S | 0.560 | 0.689 | 0.588 |
| | N | 0.704 | 1.000 | 0.632 |
| Confidenti-ality Impact | N | 0 | 0.192 | 0.200 |
| | P | 0.275 | 0.541 | 0.398 |
| | C | 0.660 | 0.662 | 0.602 |
| Integrity Impact | N | 0 | 0.196 | 0.154 |
| | P | 0.275 | 0.530 | 0.454 |
| | C | 0.660 | 0.662 | 0.611 |
| Availability Impact | N | 0 | 0.246 | 0.273 |
| | P | 0.275 | 0.438 | 0.306 |
| | C | 0.660 | 0.707 | 0.587 |

Table 3 shows confusion matrices for the CVSS prediction system before optimization, after optimizing with RR, and after optimizing with the F-measure.

*Table 2: Confusion matrices for each scoring system versions' exploit prediction*

| **CVSS equation version** | **Predicted exploit?** | **Actually exploited?** | |
|---|---|---|---|
| | | True | False |
| Un-optimized, v2.10 | True | 64 | 3433 |
| | False | 78 | 5950 |
| Risk reduction optimized | True | 142 | 9274 |
| | False | 0 | 109 |
| F-measure optimized | True | 136 | 8794 |
| | False | 6 | 589 |

Figures 3, 4, and 5 show box plots of the vulnerabilities' base score calculated with the equation before optimization, and after RR and F-measure optimization respectively. Box plots conveniently reflect the distinctness of the prediction systems' classes, and the consequently the effectiveness of the systems for exploit prediction.

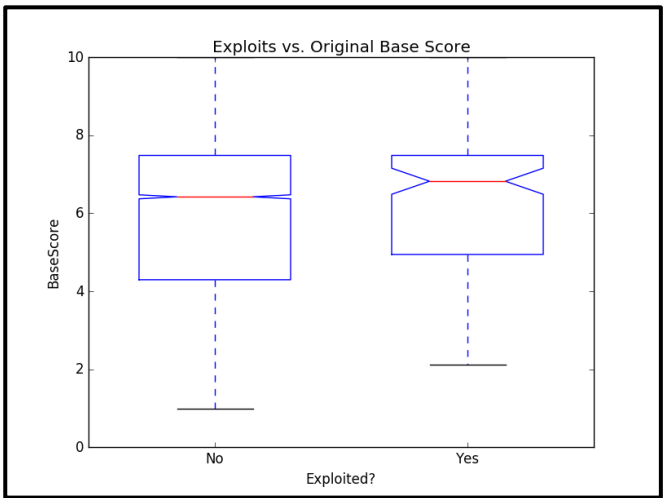


*Figure 3: Exploit prediction classification box plot before optimization for sample of 10,000 vulnerabilities*

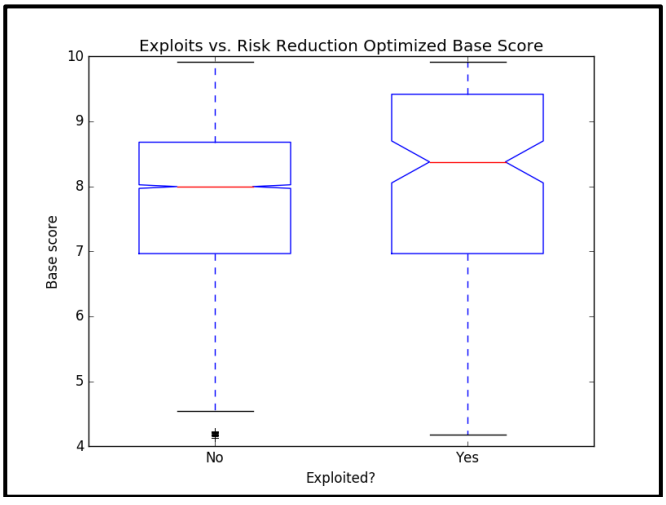


*Figure 4: Exploit prediction classification box plot after optimizing on risk reduction for sample of 10,000 vulnerabilities*

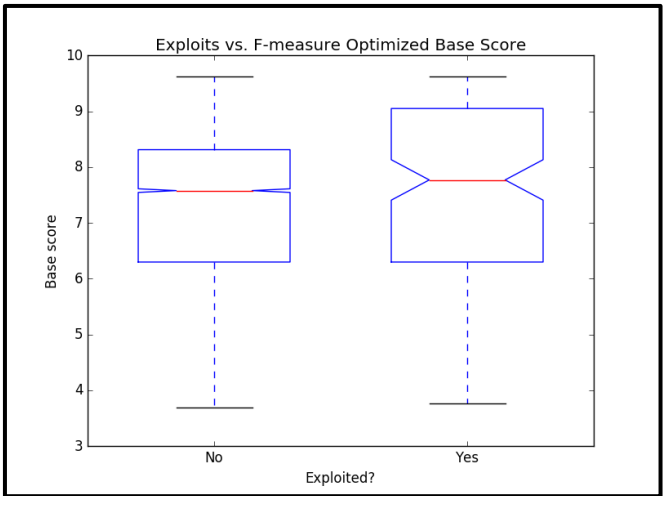


*Figure 3: Exploit prediction classification box plot after optimizing on risk reduction for sample of 10,000 vulnerabilities*

Table 4 shows the performance metrics for the CVSS scoring systems on the sample of 10,000 vulnerabilities before and after optimization with both the RR and F-measure objective functions.

*Table 3: Performance metrics before and after both optimization methods for a sample of 10,000 vulnerabilities*

| CVSS equation version | Metric | | | | |
|---|---|---|---|---|---|
| | RR | Sens. | Prec. | $ICC_{Unexp}$ | $ICC_{Exp}$ |
| Un-optimized | 0.01 | 0.45 | 0.02 | 0.50 | 0.50 |
| RR | 0.02 | 1.00 | 0.02 | 0.50 | 0.50 |
| F-measure | 0.01 | 0.96 | 0.02 | 0.50 | 0.50 |

Tables 5, 6, and 7 show the CVSS base score exploit prediction system's performance metrics when the base score equations with parameters obtained from optimization are used on 5 new samples of 1,000 vulnerabilities. P-values in Tables 6 and 7 indicate level of significance of difference with un-optimized performance. P-values could not always be calculated because of a lack of standard deviation. The program is being reviewed for errors, but the reason for constant unexploited ICC remains unclear.

*Table 4: CVSS base score performance metrics before equation optimization*

| Sample # | Metric | | | | |
|---|---|---|---|---|---|
| | RR | Sens | Prec. | $ICC_{Unex.}$ | $ICC_{Ex}$ |
| 1 | 0.03 | 0.54 | 0.07 | 0.50 | 0.57 |
| 2 | 0.07 | 0.59 | 0.11 | 0.50 | 0.55 |
| 3 | 0.03 | 0.54 | 0.07 | 0.50 | 0.50 |
| 4 | 0.03 | 0.46 | 0.08 | 0.50 | 0.54 |
| 5 | 0.09 | 0.73 | 0.12 | 0.50 | 0.59 |
| **Average** | 0.05 | 0.57 | 0.09 | 0.50 | 0.55 |
| **Standard deviation** | 0.03 | 0.10 | 0.02 | 0.00 | 0.03 |

*Table 5: CVSS base score performance metrics after optimization using RR*

| Sample # | Metric | | | | |
|---|---|---|---|---|---|
| | RR | Sens | Prec. | $ICC_{Unex}$ | $ICC_{Ex}$ |
| 1 | 0.05 | 1.00 | 0.05 | 0.50 | 0.61 |
| 2 | 0.07 | 1.00 | 0.07 | 0.50 | 0.66 |
| 3 | 0.05 | 1.00 | 0.05 | 0.50 | 0.55 |
| 4 | 0.07 | 1.00 | 0.07 | 0.50 | 0.60 |
| 5 | 0.06 | 1.00 | 0.06 | 0.50 | 0.61 |
| **Average** | 0.05 | 1.00 | 0.05 | 0.50 | 0.61 |
| **Standard deviation** | 0.01 | 0.00 | 0.01 | 0.00 | 0.04 |
| **P-value** | 0.53 | N.A. | 0.05 | N.A. | 0.03 |

*Table 6: CVSS base score performance metrics after optimization using F-measure*

| Sample # | Metric | | | | |
|---|---|---|---|---|---|
| | RR | Prec. | Prec. | $ICC_{Unexp}$ | $ICC_{Expl}$ |
| 1 | 0.05 | 0.83 | 0.07 | 0.50 | 0.58 |
| 2 | 0.07 | 0.85 | 0.10 | 0.50 | 0.53 |
| 3 | 0.00 | 0.63 | 0.05 | 0.50 | 0.50 |
| 4 | 0.06 | 0.79 | 0.09 | 0.50 | 0.53 |
| 5 | 0.06 | 0.83 | 0.09 | 0.50 | 0.55 |
| **Average** | 0.05 | 0.79 | 0.08 | 0.50 | 0.54 |
| **Standard deviation** | 0.03 | 0.09 | 0.02 | 0.00 | 0.03 |
| **P-value** | 0.94 | 0.01 | 0.43 | N.A. | 0.56 |

## 5 Discussion

The results in Tables 3 and 4 illustrate the limited improvements which can be made to the CVSS exploit prediction capabilities by optimizing its base score equation's parameters. Both optimization methods improve the system's sensitivity substantially, but have little effect on other metrics. Neither method was able to improve risk reduction. This is supported by the results from the validation trials, shown in Tables 5-7. Sensitivity improvements were significant in both cases, while optimizing on risk reduction actually resulted in a significantly lower precision (P=0.05) and exploited class ICC (P=0.03).

Consideration of the nature of sensitivity and precision may provide some explanation for this. Improving sensitivity during optimization only requires that the system not under-rate any exploited vulnerabilities; classifying every vulnerability as exploitable results in perfect sensitivity despite defeating the purpose of classification. Precision, however, requires the ability to predict exploits while it is penalized for false positives. The weighted F-measure proved insufficient to optimize the balance required by precision, while managing to improve sensitivity.

These results suggest little improvement in practical exploit prediction. This can be visualized with Figures 4-5. An effective classification system should have as much difference (variability) between different classes as possible, so they are distinct from the general population, and as little variability among classes as possible, making predications confident. Figure 3 illustrates the criticisms that the CVSS cannot discriminate between exploitable and unexploitable vulnerabilities; the classes are similar in their distribution over the base score range, with much variability among classes. Figures 4 and 5 show the same problem.

The parameter values in Table 2 may be examined in light of the lack of prediction improvements. There is a maintenance of the natural ordering of the values, and no

destruction of variability at the parameter level. The natural order refers to the tendency for the numeric values to increase as the metric attributes reflect more risk, such as partial versus complete confidentiality compromisation. Parameters converge on the same value when the optimization process is eliminating variability cause by different aspects of the equation. Since this was not observed, it suggests that the metrics used to compute the base score convey useful information, but not enough information to substantially improve exploit prediction, indicated by the lack if RR improvement. This is reflected by authors such as Khazaei *et al.* [2] and Bozorgi *et al.* [3] who had some success by extending the information considered in CVSS prediction.

The failure to improve the CVSS' prediction precision, considered with the possibility that additional vulnerability information can improve exploit prediction suggests a direction for future inquiry. Whereas authors such as Frühwirth and Männistö [6] investigated the utility of considering context-dependent, subjective Temporal and Environmental metrics, and Bozorgi *et al.* [3] and Khazaei *et al.* [2] used natural language to supplement the base score, the objectivity of the scoring system's metrics may be salvaged. Improved precision might be possible with the use of negative metrics, i.e. metrics whose cases have negative values, reflecting a reduction in risk caused by some vulnerability characteristic. More work is required to identify and validate such negative metrics.

The choice of ground-truth exploit data may be interpreted as a limitation to the validity of this study. The sole criterion for deciding whether a vulnerability was actually exploited was whether its NVD entry contained a reference to the exploit database (EDB). While the EDB was chosen to provide a wide sampling of exploit availability given its size and non-application specific intention, there are numerous exploit sources. Including additional exploit data sources provides diminishing returns given the size of the EDB, and may introduce bias if they are run by corporate interests.

## 6 Conclusions

To the best of our knowledge, this is the first work exploring the possibility of improving CVSS exploit prediction by optimizing its equation, rather than by extending the system with additional information. This was done to fulfill the original intention of the CVSS, to provide an objective and standardized vulnerability risk measure. Unfortunately, optimizing the system using risk reduction, and sensitivity and precision, made little progress to this end. Although sensitivity generally improved, false-positives increased correspondingly, negatively impacting precision. Class distinctness remained visually similar, judging by the boxplots in Figures 3, 4, and 5, and quantitatively similar as seen by the lack of substantial improvement in exploited ICCs.

Perhaps an objective and standardized vulnerability scoring system is still possible. The identification and inclusion of additional metrics in the base score equation, particularly a new variety which indicate risk reducing characteristics, are a possibility for investigation. Once the scoring metrics for an improved CVSS, or a new system entirely, are chosen, the methods presented in this work could be used to design the scoring equation. Whereas the parameters values in the CVSS may be subjective or arbitrary, choosing them based on an optimization process, if nothing else, indicates the potential of the chosen metrics.

## References

[1] - Mell P, Scarfone K, and Romanosky S. "A complete guide to the common vulnerability scoring system version 2.0" Published by FIRST-Forum of Incident Response and Security Teams. 2007.

[2] - Khazeai A, Ghasemzadeh M, and Derhami V. "An automatic method for CVSS score prediction using vulnerabilities description" Journal of Intelligent & Fuzzy Systems. 2016; 30:89-96.

[3] - Bozorgi M. Lawrence KS, Savage S, and Voelker GM. 2010 "Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits" Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. 2010; 105-114.

[4] - Younis AA, and Malaiya YK. "Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System" IEEE International Conference on Software Quality, Reliability and Security. 2015; 252-261.

[5] - Allodi L, and Massacci F. "Comparing Vulnerability Severity and Exploits Using Case-Control Studies" ACM Transactions on Information and System Security. 2014; 17(1):1-20.

[6] – Frühwirth C, and Männistö T. "Improving CVSS-based vulnerability prioritization and response with context information" International Symposium on Empirical Software Engineering and Measurement. 2009; 535-544.

[7] – Tripathi A, and Singh UK. "On Prioritization of Vulnerability Categories Based on CVSS Scores" International Conference on Computer Sciences and Convergence Information Technology (ICCIT). 2011; 692-697.

[8] – Younis AA, Malaiya YK, and Ray I. "Evaluating CVSS Base Score Using Vulnerability Rewards Programs" International Federation for Information Processing. 2016; 62-75.