# On Prioritization of Vulnerability Categories Based on CVSS Scores

Anshu Tripathi
Department of Information Technology
Mahakal Institute of Technology
Ujjain, India
anshu _ tripathi@yahoo.com

Umesh Kumar Singh
Institute of Computer Science
Vikram University
Ujjain, India
umeshsingh@rediffmail.com

*Abstract*— **In view of increasing population of vulnerabilities, quantitative evaluation of vulnerabilities is necessary for efficient mitigation. Evaluation on classified vulnerability datasets can further improve the mitigation process. Objective of this paper is to develop security metrics to prioritize vulnerability categories based on CVSS scores to step ahead in this regard. In this context, security metrics are developed to reevaluate and unify vulnerability severity scores depending on availability of patches and age of vulnerability. Proposed metrics are applied on 5177 vulnerabilities extracted from NVD published in recent one year and vulnerability categories are prioritized and ranked based on cumulative severity scores.**

***Keywords: Vulnerability, Security metric, CVSS score, Vulnerability category***

## I. INTRODUCTION

Quantitative security evaluation of risks on vulnerability datasets partitioned in well defined classes is a meaningful metric [1-4]. Quantitative evaluation also helpful in developing guidelines for allocation of resources for security testing, scheduling and development of security patches [2]. Further in [2], results of categorized vulnerability analysis shown that some vulnerability classes are more severe, this fact can be used to design optimal security solution by prioritizing severe classes. As severity level varies across categories so prioritization should be done in accordance. It'll promote better security practices by identifying most vulnerable part of system and common causes for flaws. Quantitative evaluation on categorized vulnerability data and prioritizing risk mitigation on critical classes of vulnerabilities can also result in reduced number of vulnerability scanning activity [4]. Vulnerabilities have been and are being put into various classifications, leading to a better understanding of their causes and effects and to improved vulnerability mitigation approach [3]. Considering its potential regarding better vulnerability mitigation and risk assessment proper classification of vulnerabilities is essential. Proper classification also helpful in trend analysis in order to study evolution of vulnerabilities and protecting the system proactively. There are many different vulnerability databases set up with different standards and capabilities that records vulnerabilities and classify them by several attributes. These databases serve the need of updated collection of real vulnerability data for research. Some of the most popular databases include National Vulnerability Database (NVD) [5], The Open Source Vulnerability Database (OSVDB) [6], and IBM ISS-X Force [7]. A detailed study on the contents covered and classification schemes adopted by these databases is reported in [8], on the basis of which we have selected NVD for this research work. The NVD provides a rich array of information and collaborate with standards like CVE [9], CWE [10] and CVSS [11] which makes it the vulnerability database of choice. Objective of this work is to develop a scheme to prioritize vulnerability categories that further aid in security risk analysis by strategically mitigating risks. In this work security metrics are developed to prioritize vulnerability categories based on CVSS scores. Proposed metrics applied on real vulnerability data provided by NVD and 23 vulnerability categories listed in NVD classification scheme are prioritized. The paper is organized as follows. Section II discusses related work. Section III describes proposed vulnerability category prioritization scheme. Section IV illustrates evaluation scheme applied on real vulnerability data and results presented. Section V provides comparison of results with some well known security advisories and recent vulnerability statistics reports. Finally section VI concludes the paper and presents directions for future research.

## II. RELATED WORK

Development of security metrics that quantifies objectively the risk levels due to presence of vulnerabilities requires real data. National vulnerability database [5] is one of most popular vulnerability database managed by the National Institute of Standards and Technology of the United States and is associated with the CVE [9]. It records vulnerabilities since 1999, total 46176 vulnerabilities listed under CVE names. The database specifies information under heads: ID, original release date, last revised date, source, description, impact including detailed CVSS scores, references, type and vulnerable software with versions.NVD is using CWE [10] as a classification mechanism; each individual CWE represents a single vulnerability type. All individual CWEs are held within a hierarchical structure that allows for multiple levels of abstraction. NVD uses CWEs from different levels of the hierarchical structure, by providing a cross section of the overall CWE structure. This cross section of CWEs allows analysts to score CVEs at both a fine and coarse granularity, which is necessary due to the varying levels of specificity possessed by different CVEs. There are total 23 vulnerability types in NVD classification scheme, which are based on taxonomic features vulnerability cause and vulnerability impact. Vulnerability categories are: Authentication Issues, Credentials Management, Permissions, Privileges, and Access Control, Buffer Errors, Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), Cryptographic Issues, Path Traversal, Code Injection, Format String Vulnerability, Configuration, Information Leak/Disclosure, Input Validation, Numeric Errors, OS Command Injections , Race Conditions, Resource Management Errors, SQL Injection, Link Following, Other, Not in CWE, Insufficient Information, Design Error. Last four are non CWE categories. NVD supports extensive searching under various categories, published date range, last modified date range and under different CVSS base metric parameter values. Vulnerability severity scores provided by NVD are CVSS scores. CVSS (Common Vulnerability Scoring System) [1] is a tool to quantify the severity and risk of a vulnerability to an information asset in a computing environment. It was designed by NIST (National Institute of Standard and Technology) and a team of industry partners. CVSS metrics for vulnerabilities are divided into three groups: Base metrics measure the intrinsic and fundamental characteristics of vulnerabilities that do not change over time or in different environments. Temporal metrics measure those attributes of vulnerabilities that change over time but do not change among user environments. Environmental metrics measure those vulnerability characteristics that are relevant and unique to a particular user's environment. There are six base metrics that capture the most fundamental features of vulnerability: Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrity Impact and Availability Impact (AC). The scoring process first calculates the base metrics according to the base equation, which delivers a score ranging from 0 to 10, and creates a vector. Optionally, the base score can be refined by assigning values to the temporal and environmental metrics. NVD provides CVSS base score values for vulnerabilities. Some relevant work in area of metric calculation based on CVSS score are presented here. In [12], authors evaluated quality of protection of vulnerabilities by calculating a metric based on a number of factors like the number of vulnerabilities present in the system, vulnerability history of the services and their exposure to the network and traffic patterns. Existing vulnerabilities are measured by combining severity scores of the vulnerabilities present in the system. Historical vulnerability measure for services running on the system is calculated from the frequency and severity of vulnerabilities. Decay factor used to measure historical vulnerability of the policy. Exponential averaging is applied to get unified score value as great as maximum value in data. We have used the concept of exponential decay factor in proposed scheme but arithmetic averaging applied. In [13], authors proposed a metric to rank attack patterns, our work is in line with this work. To rank attack patterns metric calculation done on dividing vulnerabilities in three equal time spans of four months each. Then, weighted sum of CVSS scores of vulnerabilities in each time span is taken and averaged. Selection of time span and value of weights are taken arbitrarily and authors have suggested modifying them based on statistics and information. The approach used in this study is different from [13], in the sense that we have divided vulnerabilities in two time spans based on availability of patches. Further, we reevaluated CVSS scores based on age of vulnerability instead of simply averaging all the scores.

## III. CATEGORY PRIORITIZATION SCHEME

In this section a set of security metrics developed to prioritize vulnerability categories unifying CVSS scores representing vulnerability severity level. Proposed metrics provide a quantitative measurement of security risk level associated with vulnerability categories. The vulnerability information is retrieved from NVD that collaborate standards: CVE, CWE and CVSS. Proposed scheme calculates severity level of a vulnerability category based on frequency and CVSS scores of vulnerabilities under it and is described below:

NVD categorize vulnerabilities in 23 categories. Each vulnerability category has a number of vulnerabilities listed under it with different severity scores presented by CVSS scores. To calculate severity level of any vulnerability category we need to calculate cumulative severity score of all the vulnerabilities listed under that category. Besides the CVSS scores there are more factors that control the impact level of vulnerabilities. Two main factors that we consider here are: first whether vulnerability has patches available to remediate it or not and second age of vulnerability. Vulnerabilities that are discovered recently and have no patches available pose more security risk as compared to vulnerabilities that have patches available. As soon as patches available not all vulnerabilities get patched immediately. With time, users patch these vulnerabilities so with the age of the vulnerability severity level decreases. Considering these factors, we developed security metric that calculate severity level of vulnerability categories and prioritize them accordingly.

To calculate severity score S(Ci), for any vulnerability category Ci; we first divided vulnerabilities under that category in two types- PU(Ci) contains those vulnerabilities that have patches not available and PA(Ci) contains those vulnerabilities that have patches available. We calculate severity scores in these two types separately and then take their weighted sum to calculate S(Ci) as formulated in equation (1).

$$S(Ci)= W1\ S(PU(Ci))+W2\ S(PA(Ci)) \quad (1)$$

Here W1 and W2 are weight factors to differentiate in risk levels posed by these two types of vulnerabilities.
In order to calculate S(Ci), we need to first calculate S(PU(Ci)) and S(PA(Ci)), severity metrics in cases patches not available and patches available respectively.

## A. *Severity metric for patches not available*

S(PU(Ci)) denotes cumulative severity score of vulnerabilities with patches not available under category Ci, calculated by formula (2) as given below.

$$S(PU(Ci)) = \frac{\sum_{j=1}^{m} CVSS\ Score(Vj)}{\sum_{i=1}^{23} CVSS\ Score(PU(Ci))} \quad (2)$$

In formula (2), m is total number of vulnerabilities with patches not available in category Ci and CVSS Score(Vj) is severity score of vulnerability j given by CVSS value. CVSS Score(PU(Ci)) is sum of CVSS scores of all vulnerabilities with patches not available in category Ci.

## B. *Severity metric for patches available*

S(PA(Ci) denotes cumulative severity score of vulnerabilities with patches available under category Ci, calculated by formula (3) as given below.

$$S(PA(Ci)) = \frac{\sum_{k=1}^{n} CVSS\ Score(Vk) * e^{-\beta\ Age(Vk)}}{\sum_{i=1}^{23} CVSS\ Score(PA(Ci))} \quad (3)$$

In formula (3), n is total number of vulnerabilities with patches available in category Ci and CVSS Score(Vk) is severity score of vulnerability k given by CVSS value. CVSS Score(PA(Ci)) is sum of CVSS scores of all vulnerabilities with patches available in category Ci. $e^{-\beta\ Age(Vk)}$ is exponential decay factor of age of vulnerability applied to reflect decrease in severity level of vulnerability with age. Age of vulnerability is difference between date of publishing and current date calculated in number of days. β is the parameter that controls rate of decay.
After calculating S(PU(Ci)) and S(PA(Ci)) from formula (2) and (3), S(Ci)- severity level of category Ci, can be calculated from formula (1). In order to prioritize categories, S(Ci) multiplied by 100 to get respective percentage of severity level of vulnerability categories as specified in formula (4) below.

$$\%S\ (Ci)= S(Ci) * 100 \quad (4)$$

## IV. EVALUATION OF SEVERITY SCORES

To prioritize vulnerability categories by metrics proposed in previous section, we have conducted an experiment on real vulnerability data extracted from National Vulnerability Database (NVD). NVD publish detailed vulnerability information from which we need vulnerability category, publishing date, patch release date and CVSS scores. Experiment is conducted on the vulnerabilities published from April 2010 to May 2011, around one year time period. Reason for selecting the one year period is that statistical studies shown that maximum time to apply patches for large organizations take time around 204 days [14]. So one year time period is fair enough to evaluate comparative severity scores of vulnerability categories and prioritize them. To conduct the experiment we need to divide vulnerability data under each category in two sets: patches not available and patches available. But NVD doesn't provide information about patch release date. Empirical studies [14, 15, 16] shown that average patch release time for vulnerability ranges between 23 to 40 days. Based on results of these studies patch release time chosen as 30 days for this experiment. Now vulnerabilities under each category divided in two time spans 01 April 2010-14 April 2011 (Patches available) and 15 April 2011- 15 May 2011 (Patches not available). Total 5177 vulnerabilities evaluated out of which 364 are with patches not available and 4813 with patches available. To calculate severity score for vulnerabilities with patches not available first sum of CVSS scores of all vulnerabilities under each category are calculated. An excerpt for category CSRF is presented in Table I. Then CVSS scores in each category are averaged on dividing by sum of CVSS scores of all 23 categories in patches not available time span (see equation (2)). Results for average CVSS scores for patches not available time span under each category are listed in column 8 of Table III, which is S(PU(Ci))
To calculate severity score for vulnerabilities with patches available, CVSS score of each vulnerability need to be recalculated with respect to its age. For this exponential decay factor need to be calculated using two parameters, age of vulnerability and β to control rate of decay. Age of vulnerability calculated in number of days is the difference between current date taken as 15 May 2011 and publishing date of vulnerability. Value of β is taken as 0.0128 based on empirical study conducted in [12].An excerpt of calculation details for category OS command injections are given in Table II. After applying exponential decay factor to CVSS score of all vulnerabilities under each category, sum of these reevaluated scores calculated as listed in last column of Table II. Then these reevaluated scores under each category are averaged on dividing by sum of reevaluated scores of all 23 categories in patches available time span (see equation (3)). Results for average reevaluated CVSS scores for patched available time span under each category are listed in column 9 of Table I, which is S(PA(Ci)).

Finally S(Ci) calculated by taking weighted sum of S(PU(Ci)) and S(PA(Ci)) with weight factors W1=0.6 and W2=0.4 . Further, value of weighted sum multiplied by 100 to calculate percentage of severity level as listed in last column of Table III. On the basis of these severity percentages, vulnerability categories ranked as specified Table IV.

TABLE I CALCULATION OF SEVERITY SCORE FOR CATEGORY CSRF FOR VULNERABILITIES WITH PATCHES NOT AVAILABLE

| CVE-ID | PUBLISHING DATE | CVSS |
|---|---|---|
| CVE-2011-1403 | 05/13/2011 | 6.8 |
| CVE-2011-1325 | 05/13/2011 | 6.8 |
| CVE-2011-1324 | 05/09/2011 | 6.8 |
| CVE-2011-1905 | 05/05/2011 | 6.8 |
| CVE-2011-1545 | 05/03/2011 | 6.8 |
| CVE-2011-1543 | 04/29/2011 | 4.3 |
| CVE-2011-1685 | 04/22/2011 | 4.6 |
| CVE-2011-1721 | 04/19/2011 | 4.3 |
| | | Sum=47.2 |
| | | S(PU(Ci)=47.2/2204=0.0214 |

TABLE II. CALCULATION OF SEVERITY SCORES FOR CATEGORY OS COMMAND INJECTION FOR VULNERABILITIES WITH PATCHES AVAILABLE

| CVE-ID | PUBLISHING DATE | CVSS score | End date | B | AGE | β *AGE | E(-β *AGE) | CVSS*E(-β *AGE) |
|---|---|---|---|---|---|---|---|---|
| CVE-2011-0456 | 03/11/2011 | 7.5 | 5/15/2011 | 0.0128 | 64 | 0.8192 | 0.4407 | 3.3058 |
| CVE-2011-0382 | 02/25/2011 | 10 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.5915 |
| CVE-2011-0381 | 02/25/2011 | 10 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.5915 |
| CVE-2011-0378 | 02/25/2011 | 8.3 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 2.9809 |
| CVE-2011-0375 | 02/25/2011 | 9 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.2323 |
| CVE-2011-0374 | 02/25/2011 | 9 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.2323 |
| CVE-2011-0373 | 02/25/2011 | 9 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.2323 |
| CVE-2011-0372 | 02/25/2011 | 10 | 5/15/2011 | 0.0128 | 80 | 1.024 | 0.3591 | 3.5915 |
| CVE-2011-0271 | 01/13/2011 | 10 | 5/15/2011 | 0.0128 | 122 | 1.5616 | 0.2098 | 2.0980 |
| CVE-2010-4278 | 12/02/2010 | 9 | 5/15/2011 | 0.0128 | 163 | 2.0864 | 0.1241 | 1.1171 |
| CVE-2010-3039 | 11/09/2010 | 6.8 | 5/15/2011 | 0.0128 | 186 | 2.3808 | 0.0924 | 0.6288 |
| CVE-2010-3757 | 10/05/2010 | 10 | 5/15/2011 | 0.0128 | 220 | 2.816 | 0.0598 | 0.5984 |
| CVE-2010-3754 | 10/05/2010 | 10 | 5/15/2011 | 0.0128 | 220 | 2.816 | 0.0598 | 0.5984 |
| CVE-2010-3753 | 10/05/2010 | 6.5 | 5/15/2011 | 0.0128 | 220 | 2.816 | 0.0598 | 0.3889 |
| CVE-2010-3752 | 10/05/2010 | 6.5 | 5/15/2011 | 0.0128 | 220 | 2.816 | 0.0598 | 0.3889 |
| CVE-2010-2445 | 07/08/2010 | 10 | 5/15/2011 | 0.0128 | 307 | 3.9296 | 0.0196 | 0.1965 |
| CVE-2010-1885 | 06/15/2010 | 9.3 | 5/15/2011 | 0.0128 | 330 | 4.224 | 0.0146 | 0.1361 |
| CVE-2010-1423 | 04/15/2010 | 9.3 | 5/15/2011 | 0.0128 | 390 | 4.992 | 0.0067 | 0.0631 |
| | | | | | | | | SUM=39.5723 |
| | | | | | | | | S(PA(Ci)=39.5723/4075.26=0.0097 |

## V. DISCUSSION

Results obtained on evaluating vulnerability data extracted from NVD in recent one year, by applying proposed scheme is shown in Table IV. Results show that top ten vulnerability categories in decreasing severity order, leaving the non-CWE categories are: Buffer Errors, Input Validation, Resource Management Errors, Cross-Site Scripting, SQL Injection, Permissions Privileges and Access Control, Numeric Errors, Path Traversal, Information leak/disclosure, CSRF. To validate this result, comparison with well known security advisories and recent security statistics reports are done. WhiteHat Website Security Statistics report 2011[17] listed top ten vulnerability classes as: Information leakage, Cross site scripting, Content Spoofing, CSRF, Brute force, insufficient authorization, predictable resource location, SQL injection, session fixation, abuse of functionality. CWE/SANS's Top 25 most dangerous software errors [18] include: Cross-Site scripting, SQL injection, Buffer errors, CSRF, Improper access control, reliance on untrusted inputs, path traversal, unrestricted upload of file, OS command injection, missing encryption of sensitive data as top ten. Veracode 2010 State of Software Security Report [19], lists back door/control channel, SQL Injection, Command Injection, Cross-Site Scripting, Insufficient Authentication, Insufficient Authorization and Remote File Inclusion as most risky vulnerability categories. The OWASP Top 10 Web Application Security Risks for 2010 [20] are: Injection, Cross-Site Scripting, Broken Authentication and Session Management, Insecure Direct Object References, Cross-Site Request Forgery (CSRF), Security Misconfiguration, Insecure Cryptographic Storage, Failure to Restrict URL Access, Insufficient Transport Layer Protection, Unvalidated Redirects and Forwards.

Comparison shows that category prioritization results corroborate with security advisories and recent statistics reports. Few reports [17, 20] focus on only web application vulnerabilities so differences are obvious. For example WhiteHat security report doesn't include buffer overflows as they don't really apply to web applications.

TABLE III CALCULATION OF SEVERITY SCORES FOR VULNERABILITY CATEGORIES

| S.No. | Category | Total No. of Vul. | No. of Vul. In PU | No. of Vul. In PA | Sum of CVSS for PU | Sum of CVSS for PA | S(PU(Ci))= Avg CVSS for PU | S(PA(Ci))= Avg CVSS for PA | %S (Ci)= Weighted Severity Score % |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Authentication Issues | 81 | 01 | 80 | 7.5 | 80.548 | 0.0034 | 0.0197 | 0.9948 |
| 2 | Buffer Errors | 706 | 37 | 669 | 317.3 | 880.48 | 0.1439 | 0.2160 | 17.2806 |
| 3 | Code Injection | 206 | 01 | 205 | 6.8 | 99.23 | 0.0030 | 0.0243 | 1.1591 |
| 4 | Configuration | 31 | 02 | 29 | 11.8 | 38.99 | 0.0053 | 0.0095 | 0.7039 |
| 5 | Credentials Management | 53 | 05 | 48 | 18.5 | 38.89 | 0.0083 | 0.0095 | 0.8853 |
| 6 | CSRF | 97 | 10 | 87 | 47.2 | 73.68 | 0.0214 | 0.0180 | 2.0081 |
| 7 | XSS | 570 | 33 | 537 | 135.2 | 237.87 | 0.0613 | 0.0583 | 6.0155 |
| 8 | Cryptographic Issues | 75 | 06 | 69 | 28.9 | 51.08 | 0.0131 | 0.0125 | 1.2881 |
| 9 | Design Error | 111 | 01 | 110 | 9.3 | 85.11 | 0.0042 | 0.0208 | 1.0886 |
| 10 | Format String Vulnerability | 19 | 0 | 19 | 0 | 23.72 | 0 | 0.0058 | 0.2328 |
| 11 | Information Leak/Disclosure | 171 | 18 | 153 | 78.7 | 82.68 | 0.0357 | 0.0202 | 2.9540 |
| 12 | Input Validation | 454 | 50 | 452 | 340.4 | 459.75 | 0.1544 | 0.1128 | 13.7796 |
| 13 | Insufficient Information | 736 | 93 | 643 | 540.5 | 449.73 | 0.2452 | 0.1103 | 19.1286 |
| 14 | Link Following | 32 | 01 | 31 | 3.3 | 31.11 | 0.0014 | 0.0076 | 0.3952 |
| 15 | Not in CWE | 01 | 01 | 0 | 4.0 | 0 | 0.0018 | 0 | 0.1088 |
| 16 | Numeric Errors | 179 | 12 | 167 | 82.3 | 178.21 | 0.0373 | 0.0437 | 3.9897 |
| 17 | OS Command Injection | 19 | 01 | 18 | 7.5 | 32.97 | 0.0034 | 0.0080 | 0.5278 |
| 18 | Other | 235 | 06 | 229 | 37.6 | 222.63 | 0.0170 | 0.0546 | 3.2089 |
| 19 | Path Traversal | 249 | 15 | 234 | 95.5 | 104.19 | 0.0433 | 0.0255 | 3.6225 |
| 20 | Permission, Privileges and Access Control | 376 | 19 | 357 | 100.7 | 282.37 | 0.0456 | 0.0692 | 5.5131 |
| 21 | Race condition | 34 | 02 | 32 | 11.1 | 22.92 | 0.0050 | 0.0056 | 0.5271 |
| 22 | Resource Management Errors | 354 | 33 | 321 | 192.7 | 378.22 | 0.0874 | 0.0928 | 8.9585 |
| 23 | SQL Injection | 388 | 17 | 371 | 127.5 | 220.89 | 0.0578 | 0.0542 | 5.6392 |
|  | SUM | 5177 | 364 | 4861 | 2204.3 | 4075.26 |  |  |  |

TABLE IV TOP TEN MOST SEVERE VULNERABILITY CATEGORIES

| Rank | CWE-ID | Category | Description |
|---|---|---|---|
| 1 | CWE-119 | Buffer Errors | Buffer overflows and other buffer boundary errors in which a program attempts to put more data in a buffer than the buffer can hold, or when a program attempts to put data in a memory area outside of the boundaries of the buffer. |
| 2 | CWE-20 | Input Validation | Failure to ensure that input contains well-formed, valid data that conforms to the application's specifications. |
| 3 | CWE-399 | Resource Management Errors | The software allows attackers to consume excess resources, such as memory exhaustion from memory leaks, CPU consumption from infinite loops, disk space consumption, etc. |
| 4 | CWE-79 | Cross-Site Scripting | Failure of a site to validate, filter, or encode user input before returning it to another user's web client. |
| 5 | CWE-89 | SQL Injection | When user input can be embedded into SQL statements without proper filtering or quoting, leading to modification of query logic or execution of SQL commands. |
| 6 | CWE-264 | Permissions Privileges and Access Control | Failure to enforce permissions or other access restrictions for resources, or a privilege management problem. |
| 7 | CWE-189 | Numeric Errors | Integer overflow, signedness, truncation, underflow, and other errors that can occur when handling numbers. |
| 8 | CWE-22 | Path Traversal | When user-supplied input can contain ".." or similar characters that are passed through to file access APIs, causing access to files outside of an intended subdirectory. |
| 9 | CWE-200 | Information Leak/Disclosure | Exposure of system information, sensitive or private information, fingerprinting, etc. |
| 10 | CWE-352 | CSRF | Failure to verify that the sender of a web request actually intended to do so. |

Four categories in NVD vulnerability classification scheme: Other, Not in CWE, Insufficient Information, Design Error are not clearly defined and have no mapping on CWE also affects evaluation results. At initial level metrics results in satisfactory ranking of vulnerability categories. In future metrics can be revised for better relevance with ranking published by security advisories.

## VI. CONCLUSION AND FUTURE WORK

Security metrics that evaluate risk levels due to presence of vulnerabilities can be highly effective in proper evaluation of system security. In this paper, we developed metrics to quantitatively evaluate severity level of vulnerability categories based on CVSS scores. To unify CVSS scores, security metric is formulated based on availability of patches and time factor. This result in evaluation and ranking of vulnerability categories objectively that can eventually lead to development of model for quantitative security evaluation of system. Results obtained on applying proposed approach are in corroboration with vulnerability category rankings published by well known security advisories and security statistics reports. For the future work metrics can be applied to more comprehensive experiments on NVD and revised.

## ACKNOWLEDGMENT

We would like to thank the anonymous reviewers who provided helpful feedback on our manuscript.

## REFERENCES

[1] Zhongqiang Chen, Yuan Zhang, Zhongrong Chen, "A Categorization Framework for Commom Vulnerabilities and Exposures." In the computer Journal Advance Access published online on May 7, 2009, http://comjnl.oxfordjournals.org,doilO.1093/comjnl/bxp040

[2] Alhazmi, O. H., Woo, S-W., Malaiya, Y. K., "Security Vulnerability Categories in Major Software Systems", Proc. Third IASTED International Conference Proceedings Communication, Network, and Information Security, 2006, pp. 138-143.

[3] Lutz Lowis, Rafael Accorsi, "On a Classification Approach for SOA Vulnerabilities," Proc. 33rd Annual IEEE International Computer Software and Applications Conference, vol. 2, 2009, pp.439-444.

[4] Somak Bhattacharya, S.K. Ghosh, "Security Threat Prediction in a Local Area Network Using Statistical Model," Proc. IEEE International Parallel and Distributed Processing Symposium, 2007, pp.425-432.

[5] NHS and NIST, National Vulnerability Database (NVD), automating vulnerability management, security Measurement, and compliance checking, http://nvd.nist.gov/scap.cfm , (Accessed on 25-05-2011).

[6] http://osvdb.org/ (Accessed on 25-05-2011).

[7] Internet Security Services Online database X-Force, 2008. [Online] Available: http://www.iss.net/xforce/ (Accessed on 25-05-2011)

[8] Tripathi, A. Singh, U.K., "Taxonomic Analysis of Classification Schemes in Vulnerability Databases" (Communicated)

[9] Common Vulnerabilities and Exposures. [Online]. Available:http://cve.mitre.org (Accessed on 25-05-2011)

[10] Common Weakness Enumeration. [Online]. Available: http://cwe.mitre.org (Accessed on 25-05-2011)

[11] Forum Of Incident Response And Security Teams (FIRST), "Common vulnerability scoring system 2.0," 2007. [Online]. Available: http://www.first.org/cvss (Accessed on 25-05-2011)

[12] Muhammad Abedin, S. Nessa, E. Al-Shaer and L. Khan, "Vulnerability analysis for evaluating quality of protection of security policies", QOP'06: proceedings of the 2$^{nd}$ ACM workshop on Quality of protection, 2006, pp. 49-52.

[13] J. A. Wang, H.Wang, M.Guo, L.Zhou and J. Camargo, "Ranking Attacks Based on Vulnerability Analysis", In Proc. 43$^{rd}$ Annual Hawaii International Conference on System Sciences, 2010, pp. 1-10.

[14] Y. Beres, J. Griffin, S. Shiu, M. Heitman, D. Markle, and P. Ventura, "Analysing the Performance of Security Solutions to Reduce Vulnerability Exposure Window", in Proc. ACSAC, 2008, pp.33-42.

[15] Kemal Altinkemer, Jackie Rees and Sanjay Sridhar, "Vulnerabilities and Patches of Open Source Software: An Empirical Study", In the Journal of information System Security, Volume 4, Number 2, 2008, pp. 3-25.

[16] A. Arora, R. Krishnan, R. Telang, and Y. Yang, "An Empirical Analysis of Software Vendors' Patch Release Behavior: Impact of Vulnerability Disclosure", presented at Information Systems Research, 2010, pp.115-132.

[17] WhiteHat Website Security Statistics Report – Winter 2011, 11th Edition
Measuring Website Security: Windows of Exposure, 2011, [Online]. Avaialble: https://www.whitehatsec.com/home/resource/stats.html, (Accessed on 25-05-2011)

[18] CWE/SANS TOP 25 Most Dangerous Software Errors. [Online] Available: http://www.sans.org/top25-software-errors, (Accessed on 25-05-2011)

[19] VERACODE STATE OF SOFTWARE SECURITY REPORT: VOLUME 2, 2010, [Online]. Available: http://www.veracode.com/reports/index.html, (Accessed on 25-05-2011)

[20] OWASP Top Ten Project. Open Web Application Security Project. [Online]. Available: http://www.owasp.org/index.php/Category:OWASP Top Ten Project, (Accessed on 25-05-2011)