International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

IVSV: An Improved CVSS Base Score Mechanism with Vulnerability Type

Gagandeep Chawla, Neeraj Sharma, Narender Kumar Rawal

Abstract: ***Increased demand of Software and Applications offer intruders to perform malicious activities and exploit user's personal data. Ignorance of security measures and tools while coding the software promotes the vulnerabilities and flaws. Developing a secure and bug free software is a big challenge for a developer and needs proper attention towards safety features. A single security mistake can lead to a loss of important information or confidential business data. Software companies and other organizations are looking for improved vulnerability security systems to narrow down the risk of vulnerabilities. Risks like social security attacks, bugs, phishing emails, vulnerabilities, virus attacks and more, hover over the IT industry. Threats are possible from all directions and in many different ways, so having an adequate vulnerability scoring mechanism is highly needed to reduce the risk of attacks. Identifying these threats before they get close enough to do damage is the most practical way to handle them. CVSS-V2 (Common Vulnerability Scoring System) is a standard for scoring the severity of vulnerabilities. CVSS-V2 uses three equations (Base, Temporal and Environmental) to capture and rate vulnerability severity. Numerous IT companies and government organizations rely on CVSS to evaluate and prioritize vulnerabilities. This paper proposes a method as an improvement over CVSS-V2 scoring system by introducing "Vulnerability type" in its base score equation.***

Keywords: CVSS-V2, Vulnerability type, IVSV, NVD

I. INTRODUCTION

Software Vulnerabilities and bugs are the root cause of security issues in software and requires immediate attention in order to avoid loss. There has been a rapid rise in the number of vulnerabilities and cyber-attacks on the Internet in recent years [1]. Recognizing a bug or vulnerability quickly is a matter of great concern for developing bug free software. Software developers have to prioritize these bugs and need to remediate them as early as possible [2]. Implementation of certified security tools and an alert security team can raise the graph of successful software development. However, the burden and pressure to complete software projects on time promote negligence of security checks. To overcome these issues, security organizations have developed numerous scoring schemes which cut down the risk of vulnerabilities in software and applications. CVSS (Common Vulnerability Scoring System) empowered by NIAC (National Infrastructure Advisory Council) is an open tool that is of great help to the industry to overcome such security issues. CVSS consists of three metric groups. These are Base, Temporal and Environmental Metric groups. Of these, the latter two i.e. Temporal and Environmental are optional. Base metric group of CVSS-V2 basically holds characteristics that do not change over time. Temporal metric contains the features of vulnerabilities that change over time. Environmental metric contains the features of vulnerabilities that are dependent on user's operating environment [3].

After extensive literature review it has been found that none of researcher dealt with "Vulnerability type" which is an important factor to be considered in the base score equation. Spanos et al. in [4] proposed an approach for scoring vulnerabilities wherein they used the formula *"Score = round to 1 decimal (Exploitability Score + Impact Score) * f(Impact)"* to compute the vulnerability score and recommended to use 'vulnerability type' to enrich the vulnerability scoring.

Risk associated with vulnerabilities can be better assessed by assessing the type of vulnerability. CVSS approach is the de facto standard used all around the world to evaluate the severity of vulnerabilities [4]. CVSS exploitability measure is criticized as it assigns static numbers based on expert knowledge without considering the vulnerability type, [5]. To overcome this drawback, it is very important to consider 'vulnerability type' into CVSS base score equation.

The objective of this paper is to propose a new base score formula which concludes better results while considering vulnerability types. Four most common vulnerability types are included in proposed base score formula including 'buffer overflow', 'race condition', 'unvalidated input', and 'authentication weakness'. These vulnerability types are given reference values 1.0/0.8/0.6/0.5 subject to their severity levels. The highest value 1.0 is given to buffer overflow as it is the most common error found in software while coding.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
Gagandeep Chawla, Computer Applications, Punjab Technical University, Jalandhar, India.
Dr. Neeraj Sharma, Management & Computer Applications, GJIMT, Mohali, India
Dr. Narender Kumar, Computer Science & Engineering H.N.B Garhwal University, Srinagar, India.

II. VULNERABILITY TYPES

Computer vulnerability refers to a glitch that causes the system to make an unwanted action. In order to build bug free software, it is expected to have proper understanding of different vulnerability types. This section presents a brief introduction to some common and important types of vulnerabilities.

(i) Buffer Overflow

It is a coding mistake in which an application writes data past the end or at starting of a buffer. This occurs when writable data is more than the capacity of a buffer. This leads to writing of data at the adjacent storage of the memory. This overflow causes applications to compromise data and moreover create an entry point for attackers. Buffer overflow arises due to lack of boundary check in C functions [6]. Standard library functions should be avoided by the developers as they are not bound checked. Fig 1 shows, situation (a) when main program is running, situation (b) shows when program 'P1' is called and situation (c) shows buffer overflow in grey [7].

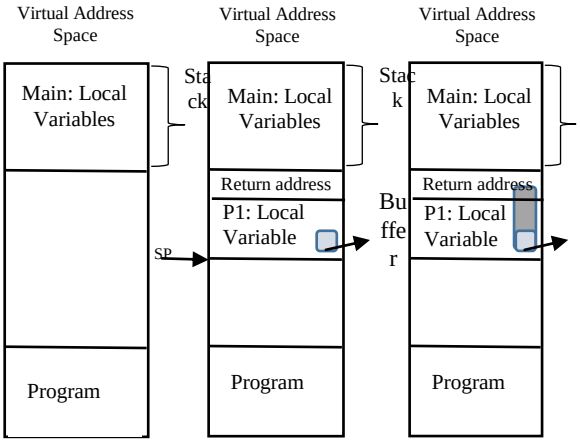

Fig 1: Buffer over flow shown in grey

(ii) Race condition

It is also called concurrency attack or thread jacking. It is a situation which normally occurs in logic gates when certain input comes into conflict. This is a result of interferences that occur due to multiple threads running in the computer and are trying to share same resources [8]. Race condition often leads to serious vulnerability conditions. This is one of the most common problem experienced while writing multithreaded applications. Race condition also occurs in a network where two or more users try to access a resource at same time. Fig. 2 shows the race condition when some processes try to access a shared resource.

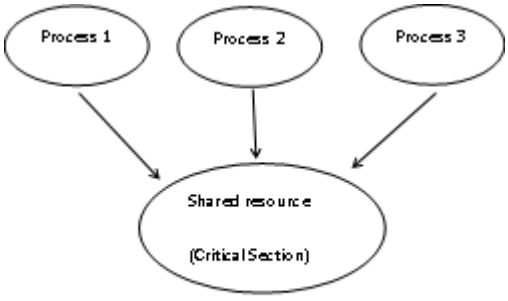

Fig 2: Process sharing same

(iii) Unvalidated Input

It is a weakness mostly found in web applications which users input from http requests. After controlling http requests, attackers can bypass the site's security measures. Common unvalidated attacks include- Command insertion, cross site scripting, SQL injection etc. Websites try to protect them by filtering out the malicious inputs. Vulnerability may occur if a program accepts and uses unvalidated input to construct a dynamic SQL query to SQL database [9]. Fig. 3 describes a situation where a Server exchanges data in both directions i.e. towards database and clients through HTML. The approach to handle unvalidated input must be unique as per the development environment.

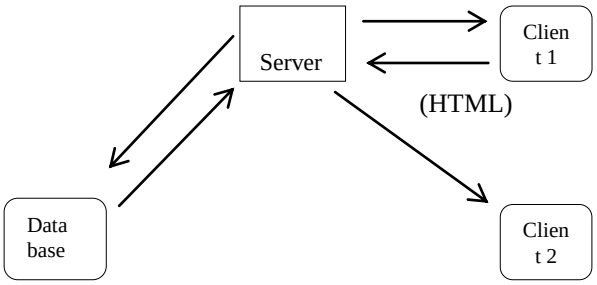

Fig 3: Server exchanging data towards database and clients

(iv) Authentication Weakness

Weak credentials are often the weakest link in most authentication systems. By improving the passwords, users can make it difficult for hackers to enter into the system [10]. Keeping stronger and tough passwords is required as weak passwords can result in a compromise with the important information. Since passwords are normally the only access requirement, it's worth putting some attention in keeping the passwords secure. Brute force is a common technique adopted by intruders to apply on user's system for guessing the possible password. Users should avoid using common passwords like "1234…", "admin", "nick name" etc. which are easy to hack. Even the passwords should be changed regularly at specified interval of time as a safety measure.

III. IVSV: A NEW IMPROVED VULNERABILITY SCORING SYSTEM WITH "VULNERABILITY TYPE"

Although CVSS is a de facto standard and carry many features but after a detailed literature study it is found that the "vulnerability type" is an important factor to be considered in CVSS base score equation (1). In this section, we propose a new method IVSV (Improved Vulnerability Scoring System with "Vulnerability Type").

Proposed IVSV improves the scoring mechanism by introducing "Vulnerability Type" and making a selection out of four subcategories of vulnerability types i.e. 'Buffer overflow', 'Race condition', 'Unvalidated input' and 'Authentication weakness'. These subcategories are the most common type of vulnerabilities found in software during normal coding process and requires equal attention as is required by other vulnerabilities.

Original CVSS-v2 base score equation (1) contains six metric namely 'Access Vector', 'Authentication', 'Integrity Impact', 'Access complexity', 'Confidentiality Impact' and 'Availability Impact' as shown in fig. 4. In the proposed IVSV, this metric group is improved to include a total of seven metric groups. This includes the six groups from the CVSS-v2 base equation (1) and an additional new metric as "Vulnerability type" which is introduced in the proposed IVSV. Fig. 5 shows the metric groups of the proposed IVSV.

Most software vulnerabilities fall into one of these metrics. Outcomes are practically checked on a sample of vulnerabilities from a database called NVD (National Vulnerability Database).

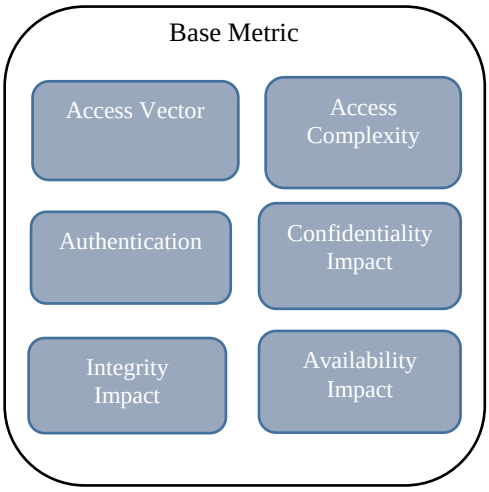

Fig 4: CVSS-v2 metric groups

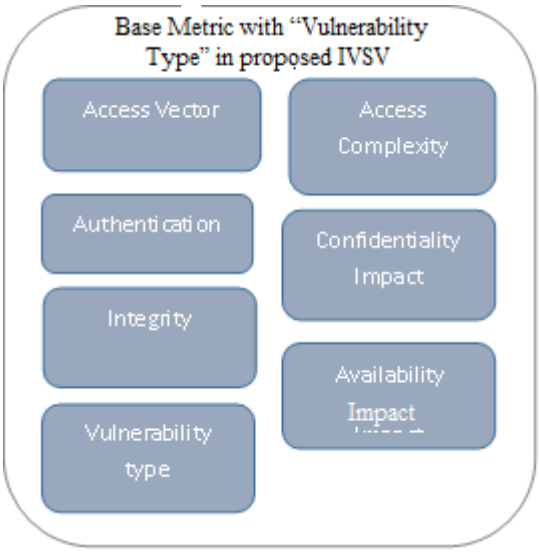

Fig 5: Base Metrics in proposed IVSV

The equations for calculation of vulnerability scores in CVSS-v2 are shown in equation 1, equation 2, equation 3 and equation 4. Most of the times the software developers and companies rely on the base score itself.

Original base score equation of CVSS-v2

Base Score=round to One decimal (((0.6*Impact) + (0.4*Exploitability)-1.5)*f(Impact)) ..... (1)

Impact=10.41*(1-(1-CI)*(I-II)*(I-AI) ..... (2)

Exploitability = 20*AV*AC*Au ..... (3)

f(impact)=0 if impact=0, 1.176 otherwise.....(4)

Equation 1, equation 2, equation 3 and equation 4 shows the original base score equations of CVSS–v2. In these equations *CI* refers to 'Confidentiality Impact', *II* refers to 'Integrity Impact', *AI* refers to 'Availability Impact', *AV* refers to 'Access Vector', *AC* refers to 'Access Complexity' and *Au* refers to 'Authentication'.

The proposed IVSV computes the vulnerability score by modifying the CVSS-v2 base score equation (1) to accommodate the new introduced parameter *'vulnerability type'*. Equation 5, equation 6, equation 7, equation 8 and equation 9 shows the modified equations in the proposed IVSV. Equation 8 computes the weight component for the final vulnerabitiy score with the introduced parameter 'Vulnerability Type' in the proposed IVSV.

IVSV Base Score Equation

Base Score= Round to one decimal (((x*Impact) + (y*Exploitability) + (z*Vulnerability type)-1.5)* f (Impact))..... (5)

Impact = 10.41*(1-(1-CI)*(1-II)*(1-AI)) ..... (6)

Exploitability = 20*AV*AC*Au..... (7)

Vulnerability Type = 10*VC..... (8)

f(impact)= 0 if Impact=0, 1.176 otherwise.....(9)

In equation (5), 'x' represents the weight value associated with Impact factor. 'y' represents the weight value associated with exploitability factors and 'z' represents the weight value associated with vulnerability type.

Table 1: Metric Factors for Vulnerability Type

Metric	Description	Metric Value	Reference Value
VC	Type of Vulnerability	Buffer overflow/ Race condition/ Unvalidated Input/ Authentication Weakness	1.0/0.8/0.6/0.5

As per study and knowledge, values of x, y and z are adjusted are adjusted to 0.5, 0.3 and 0.2. Influence weight of impact factor is adjusted to 10.41 accounts for the base metric.

CI refers to Confidentiality Impact, *II* refers Integrity Impact and *AI* refers Availability Impact. These three have three metric references values none, partial and complete. The "vulnerability type" calculation is shown in equation (8). Where *VC* refers the 'vulnerability category.

IV. RESULTS AND DISCUSSION

In this section, the proposed scoring system IVSV is applied on a sample set of real vulnerabilities taken from National Vulnerability Database (NVD) and results are compared with original CVSS-v2 scores. Table 2 shows the scores resulted after including "vulnerability type" and original base score computed with base score formula of CVSS-v2. Obtained results shows that the proposed system (IVSV) has a significant impact on scores calculated with original base score equation. The vulnerability scores of range between (8-1.0) are considered as high; (4-7.9) are considered as medium and (0-3.9) are considered as low.

We have given equal importance to impact and exploitability factors of base score equation by setting their weights to 0.5 and 0.3 respectively and adding on one extra factor vulnerability type with weight value 0.2.

TABLE 2: Computed Base Scores of IVSV

Vulnerability ID	Original Base Score CVSS	Scores resulted with Vulnerability type			
		Buffer	Race	Unvalidated Input	Authentication
CVE-2017-18015	4.3	5.3	4.8	4.4	4.1
CVE-2017-9964	5.8	6.5	6.1	5.6	5.3
CVE-2018-3814	6.5	7.2	6.7	6.2	6.0
CVE-2017-1000432	6.0	6.8	6.3	5.8	5.6
CVE-2018-6476	10	10	9.5	9.1	8.8
CVE-2018-4837	5.0	5.8	5.3	4.9	4.6
CVE-2018-6205	6.1	7.0	6.5	6.1	5.8
CVE-2018-18077	5.0	5.8	5.3	5.3	4.6
CVE-2018-4836	6.5	7.2	6.7	6.2	6.0
CVE-2018-6029	5.0	5.8	5.3	4.9	4.6

After comparing the results with original base score it has been observed that there is a slight increase in 'Buffer overflow' and 'Race condition' type vulnerabilities and slight decrease in 'Unvalidated Input' and 'Authentication weakness' type vulnerabilities. The reason is 'buffer overflow' and 'race conditions' are the most common faults done by the coders or software professionals while developing the software.

For example, the first vulnerability CVE-2017-18015 from Table 2, the original base score is 4.3 and scores resulted with vulnerability types are:

Buffer overflow=5.3
Race condition=4.8
Unvalidated input=4.4
Authentication weakness=4.1

As discussed above buffer overflow and race condition are more prone to errors, so score obtained is 1.0 higher for 'buffer overflow' and 0.5 higher for 'race condition'. There is 0.1 point difference in 'unvalidated input' and decrease of 0.2 points for 'authentication weakness'.

Roughly the resulted scores are around the original CVSS-v2 base scores and show that the addition of vulnerability type has impact on scoring vulnerabilities. Even though temporal and environmental metrics are optional but adding vulnerability type overall CVSS score will vary.

V. CONCLUSION

This paper identified a new modified assessment technique to score the severity level of vulnerabilities. New proposed technique considers a new feature "Vulnerability type" as an important factor to be included in conventional CVSS-v2 base score equation. 'Vulnerability type' provides a clear image about the type of bug which has affected an application or software and helps to achieve more authentic and accurate score. Vulnerability type is further subcategorized into four most common categories. These subtypes are 'buffer overflow', 'race condition', 'unvalidated input' and 'authentication weakness'. Table 3 shows that the CVSS-v2 base score equation and proposed IVSV base score equation.

Table 3: CVSS base score equation and IVSV equation

CVSS-v2	BS=round to One decimal (((0.6*Impact) + (0.4*Exploitability)-1.5)*f(Impact)
IVSV	Base Score= Round to one decimal (((x*Impact) + (y*Exploitability) + (z*Vulnerability type)-1.5)* f (Impact))

Proposed equation is tested on sample set of real vulnerabilities database taken from National Vulnerability Database (NVD). By comparing the results with conventional CVSS-v2 results it has been observed that 'vulnerability type' do make a difference in calculating scores. As an outcome, a new advanced and significant scoring model is obtained which helps in selection of most critical vulnerabilities.

The main objective to propose this new approach is to overcome the weakness of CVSS-v2. This approach will support programmers and coders to line up more severe vulnerabilities.

REFERENCES

1. Laurent Gallon LIUPPA, "Vulnerability discrimination using CVSS framework", IEEE 978-1-4244-8704-2/11, 2011
2. Ayala Goldstein, "https://resources.whitesourcesoftware.com/blog-whitesource/3-essential-steps-for-your-vulnerability-remediation-process" July 2018.
3. Peter Mell, Karen Scarfone, Sasha Romanosky "A Complete Guide to the Common Vulnerability Scoring System Version 2.0" June 2007.
4. Georgios Spanos, Angeliki Sioziou, Lefteris Angelis "WIVSS: A New Methodology for Scoring Information Systems Vulnerabilities" PCI, Sep. 2013.
5. M. Bozorgi, L. K. Saul, S. Savage, & G. M. Voelker, "Beyond heuristics: learning to classify vulnerabilities and predict exploits," USA, ACM, July 2010.
6. Gary Mullen, Liam Meany Entwine, "Assessment of Buffer Overflow Based Attacks on an IoT Operating System" School of Electronic Engineering, IEEE, 2019.
7. Abhinav Jain "https://www.quora.com/What-is-a-buffer-overflow" 2018.
8. Tanjila Farah, Rashed Shelim, "Study of Race Condition: A Privilege Escalation Vulnerability" Research Gate, June 2017.
9. William L. Fithen "Never Use Unvalidated Input as Part of a Directive to any Internal Component" June, 2013.
10. Affinity "https://affinity-it-security.com/what-is-weak-authentication/"

AUTHORS PROFILE

Gagandeep Chawla is research scholar in computer applications and has obtained MCA degree in Computer Applications from Punjab Technical University, Jalandhar. He is MCSE-2000 (Microsoft Certified System Engineer). He has presented more than 10 research papers in National and International conferences and has attended more than 10 workshops and seminars. He carries more than 14 years of experience in teaching and industry and around 6 years of experience in research. His research interests include Advance Software engineering, Vulnerability scoring system and computer networks. He has worked as Assistant to Controller to conduct Punjab Technical University final examination.

Dr. Neeraj Sharma is Ph.D., MCA, MBA, and M.SC degrees from Punjab University, Patiala and Kurukshetra University, Kurukshetra. He has published more than 20 research papers in National and International conferences. He has conducted and coordinated various Training and Faculty Development Programs. He is having 20 years of rich experience in varied fields including Teaching, Administration, Marketing, Software development with consistently increasing responsibilities. Currently; he is working as Dean with the Gian Jyoti Institute of Management and Technology, Mohali. He is acting as Supervisor for eight students of Ph.D. on various research areas of Management and Computer Applications.

Dr. Narender Kumar is Ph.D. from IIT, Roorkee. He has published more than 15 research papers. His main research work focuses on Cryptography Algorithms, Software Engineering, Data Mining, Genetic Algorithm, PSO and Digital Image Processing. He is currently working as Assistant Professor in the Department of Computer Science and Engineering at H.N.B Garhwal University, Srinagar since 2012. He is the founder coordinator of e-governance System of the University from last 5 years. He has 11 years of teaching experience and 6 years of Research Experience.