International Journal of Mathematical, Engineering and Management Sciences
Vol. 5, No. 2, 328-342, 2020
https://doi.org/10.33889/IJMEMS.2020.5.2.027

Assessment of Software Vulnerabilities using Best-Worst Method and Two-Way Analysis

Misbah Anjum
Amity Institute of Information Technology,
Amity University, Noida, Uttar-Pradesh, India.
Corresponding author: bhatmisbah6@gmail.com

P. K. Kapur
Amity Center for Inter-Disciplinary Research,
Amity University, Noida, Uttar-Pradesh, India.
E-mail: pkkapur1@gmail.com

Vernika Agarwal
Amity International Business School,
Amity University, Noida, Uttar-Pradesh, India.
E-mail: vernika.agarwal@gmail.com

Sunil Kumar Khatri
Amity University, Tashkent Campus, Tashkent, Uzbekistan.
E-mail: sunilkkhatri@gmail.com

(Received March 21, 2019; Accepted August 21, 2019)

Abstract

Software is one of the most essential part in today's world, with its requirements in every industry be it automotive, avionics, telecommunication, banking, pharmaceutical and many more. Software systems are generally a bit complicated and created by distinct programmers. Usually any mistake in the code by a programmer in the developing stage of a software can lead to loopholes that cause vulnerabilities. Vulnerability is a software flaw that an assaulter can exploit to conduct unlawful activities within a computer system. Despite the understanding of vulnerabilities by the academia and industry, the amount of vulnerabilities is growing exponentially as fresh characteristics are added to the software frequently. Developers and testers are faced with the challenge of fixing large amounts of vulnerabilities within limited resources and time. Thus, prioritizing software vulnerabilities is essential to reduce the usage of corporate assets and time, which is the motivation behind the present study. In the present paper, the issue of software vulnerability prioritization is addressed by utilizing a new multi-criterion decision-making (MCDM) technique known as the Best Worst method (BWM). Further, to assess the vulnerabilities in terms of their critical nature, we have applied Two-Way assessment technique. The BWM utilizes two pairwise comparison vectors to determine the weights of criteria. The two-way assessment framework takes into account the perspectives of both managers/developers and stakeholders/testers to highlight the severity of software vulnerabilities. This can act as a significant measure of efficiency and effectiveness for the prioritization and evaluation of vulnerability. The findings are validated with a software testing firm from North India.

Keywords- Software vulnerability, Multi-criteria decision making (MCDM), Best worst method (BWM), Two-way assessment technique.

1. Introduction

The world progresses at an agile pace of technology and complexity as we know it today. The world is progressing at an agile rate in technology and complexity. The internet has developed as a necessity by the virtue of continuous developments in networking. Almost all our day-to-day activities involve the use of computers. Computers are integrated into wristwatches, mobile phones, household appliances, buildings, vehicles and aircrafts (Lyu, 1996). Even when we look at any sector, they are extremely dependent on computers for their fundamental functioning. This increasing dependence on computer systems has further highlighted the current security problems in the software world (Kapur et al., 2011). Security of these systems thus, is very important, because any violation of information security poses an increasing and serious risk to global security and individual economic well-being (Arora et al., 2010). In 2017, the overall increase in the reported vulnerabilities was 13% and the vulnerabilities associated with the Industrial Control System (ICS) increased 29%, compared to 2015 and 2016, according to the Symantec Internet security threat report (US, 2018). These data breaches often exploit vulnerabilities in the software. The percentage of vulnerabilities found and revealed has thus, increased dramatically over the past few years (Liu et al., 2012). There were 6,787 vulnerabilities disclosed in 2014, compared with 5,291 in 2013 as per the reports of Symantec Internet Security threat report (US, 2013)

A software vulnerability can be perceived as system defect, weakness, or even a system mistake that an assailant can exploit to change system behavior (Jimenez et al., 2009). It is the responsibility of the security team to recognize and solve these vulnerabilities through various software and hardware platforms (Kapur et al., 2014). These vulnerabilities have to be prioritized and evaluated to comply with company deadlines and operate within restricted budgetary resources.

Assessment should be done in a manner that first those vulnerabilities should be resolved which pose the biggest threat (Sharma et al., 2019). Prioritization of vulnerabilities include various characteristics that developers and testers need to consider when choosing the order to fix the vulnerability. Therefore, a very significant task for developers and testers is to define these vulnerabilities based on their severity so that they can be handled properly according to the level of their damage causing capabilities and a timely patch can be released. A crucial vulnerability is one which, if exploited by attackers, may allow malicious code to be executed without user interaction, possibly leading to the security breach (US, 2013). Number of qualitative and quantitative rating methods have been studied in the literature to assign scores to vulnerabilities (Liu and Zhang, 2011). A variety of vulnerability scores have been established, supported and implemented by a various computer and non-profit providers and organizations, to assess the danger qualitatively (e.g. X-Force, 1999; Symantec Corporation, 2000; Microsoft, 2002; Secunia; VUPEN, 2005; Red-hat; Mozilla; Google) or quantitatively (e.g. CERT; CVSS, 2007). In the previous studies of software vulnerability evaluation, prioritization has been done using various Multi-Criteria Decision-Making (MCDM) techniques for example, Sibal et al. (2017) prioritized software vulnerabilities using Analytic Hierarchy Process (AHP), Normalized Criteria Distance (NCD) and Decision Making Trial and Evaluation Laboratory (DEMATEL). Liu and Zhang (2011) proposed Verbal Rating Scales (VRSS) based software vulnerability prioritization using AHP. Huang et al. (2013) evaluated software vulnerability prioritization using Fuzzy Analytical Hierarchy Process (FAHP) and fuzzy synthetic decision making approach. Kansal et al. (2017) prioritized vulnerabilities using Analytic Network Process (ANP) method. Hence, no work has been done in the vulnerability prioritization using freshly developed MCDM approach called as best worst method (BWM) (Rezaei, 2016). In this paper, we propose the best worst method to rank and prioritize the vulnerabilities and also two-way assessment analysis is done to calculate the overall criticality measure of vulnerabilities.

1.1 Research Objectives

Overall, from the above discussion the following research objectives can be identified:

- To identify the software vulnerability types for prioritization and assessment, from industry professionals, academicians and literature review.
- To rank the vulnerabilities using the BWM approach; and
- Evaluate identified vulnerabilities with a two-way assessment strategy in relation to their criticality.

To achieve the study goals, a two phase methodology comprising of BWM and two-way analysis approach is used. We first ascertain the priority of vulnerabilities in terms of their weights. Secondly, two-way assessment technique is used to evaluate vulnerabilities in terms of their critical nature.

The paper is organized as follows: Section 2 discusses the research methodology. Section 3 represents the numerical illustration followed by results discussion in section 4. Section 5 provides a short overview on the conclusion of the article and the future work feasible.

2. Research Methodology

The current section focuses on the research methodology. In this study, we have presented a two-fold methodology to evaluate multiple software vulnerabilities. The vulnerabilities are prioritized in first phase using BWM. In second phase, we have used two-way assessment technique to calculate the overall criticality of vulnerabilities based on the weights collected from BWM. Once we get the priorities of software vulnerabilities by BWM, we analyze their seriousness by using two-way assessment technique in the second phase. Two-way analysis takes into consideration both the developers as well as testers perspective simultaneously.

2.1 Dataset Description

Initially, before the study starts, the literature survey identifies several kinds of software vulnerabilities. (National Vulnerability Database) and CVE details (Özkan, 1999) are the authentic sites for data collection. As the aim of this research is the assessment and classification of software vulnerabilities for their harmful capacity (criticality), this study targets industry experts (managers/developers and testers/stakeholders) from one of the North Indian software company. A panel of experts was created to discuss finalization of types of vulnerabilities. A consensus panel strategy usually demonstrates that several experts ' opinions are always better than a single expert's view. The panel consists of experts having minimum experience of 8-10 years in the respective fields of development and testing.

2.2 Delphi Method

Delphi method have been used to collect the data. The Delphi method is a procedure used by an expert panel to achieve a group view or decision (Okoli and Pawlowski, 2004). Steps involved in Delphi method are as follows:

- Experts answer multiple questionnaire rounds and after each round the answers are gathered and shared with the group.
- After every round, specialists can change their responses based on their interpretation of the Group reaction.
- The end outcome has to be a real consensus about what the group believes.

Through the discussions with the expert panel discussions, a total of nine (9) vulnerabilities were finalized which are discussed in detail in Table 1.

Table 1. Description of software vulnerabilities with references

Notations	Vulnerability	Definition	Reference
SV1	SQL Injection (SQLI)	In SQLI an attacker submits the code as input in such a way that it executes against the database. It is database focused and is used to steal data from databases.	Khurana et al. (2017)
SV2	Cross Site Scripting (XSS)	It refers to the family of attacks where attackers execute their code in the browsers of your website visitors. It aims towards attacking the end user.	Liu et al. (2012)
SV3	Buffer overflow (BO)	It occurs when more data is put into a fixed length buffer than the buffer can handle. This extra information can flow to an adjacent memory space, corrupting or overwriting data in that space.	Huang et al. (2013)
SV4	Cross Site Request Forgery (CSRF)	Web application does not sufficiently verify whether a well formed, valid and consistent request was intentionally provided by user who submitted the request.	Liu et al. (2012)
SV5	File Inclusion (FI)	It refers to the inclusion of functionality from untrusted control sphere.	Sibal et al. (2017)
SV6	Information Gain (IG)	It is the intentional or unintentional disclosure of information to an actor that is not explicitly authorized to have access to that information.	Narang et al. (2017)
SV7	Code Execution (CE)	It occurs when the output or content served from a web application can be manipulated in such a way that it triggers server side code execution.	Sibal et al. (2017)
SV8	Gain of Privileges (GP)	Weakness in this category are related to the management of permissions, privileges that are used to perform access control.	Sharma et al. (2019)
SV9	HTTP Response Splitting (HTTPR)	Data enters a web application through http request. The data is included in HTTP response header sent to a web user without being validated for malicious characters.	Ozkan, (1999)

2.3 Best Worst Method

To select from the original list for the selection of most prominent vulnerabilities, we have used BWM (Rezaei, 2015). For this purpose, we have consulted five senior DMs (with 10 years of experience) of the software testing company situated in national capital region (NCR), India. The BWM is utilized to generate weights of the software vulnerabilities using least comparisons, which is the highlight of this method over other MCDM techniques. The method only requires two the comparison vectors of the best with other vulnerabilities and the other with worst factor. This decreases the decision-making time and complexity. Let the chosen vulnerabilities sets be $SV = \{SV_1, SV_2, ..., SV_9\}$.The decision making process of BWM for prioritization of vulnerabilities comprises of the following steps (Rezaei, 2015; Govindan et al., 2019).

Step 2.3.1: Selection of best and the worst criteria.
The DMs select the best/most critical and the worst/least criteria.

Step 2.3.2: Calculate the preference of best criteria over the other criteria set *C*.

To calculate the preference of the best criteria over the other criterias, the DMs use a score of 1-9. The resulting vector of “Best-to-Others” is:

$$V_B = \left(v_{B1}, v_{B2}.., v_{Bn}\right)$$

where, v_{Bi} refers to the numerical importance of the best criteria *B* over *i*th attribute and $v_{BB} = 1$.

Step 2.3.3: Calculate the preference of the criteria set *C*, over the worst criteria.
In this step, we use the score of 1-9 to calculate the preference of the others over worst criteria. The resulting vector of “Worst-to-Others” is:

$$V_W = \left(v_{1W}, v_{2W}.., v_{nW}\right)^T$$

where, v_{iW} gives the preference of the i^{th} criteria over worst criteria *W* and $v_{WW} = 1$.

Step 2.3.4: Determine the optimal weights of criteria.
The aim of this step is to calculate the optimal weighting vector $\left(x_1^*, x_2^*..., x_n^*\right)$ of the criteria.
The optimal weight of i^{th} criteria is the one which meets the following requirements:

$$\frac{x_B^*}{x_i^*} = v_{Bi} \quad \text{and} \quad \frac{x_i^*}{x_W^*} = v_{iW} .$$

In order to satisfy this condition, we need to minimize the maximum absolute difference $\left|\frac{x_B}{x_i} - v_{Bi}\right|$ and $\left|\frac{x_i}{x_W} - v_{iW}\right|$ for all criteria.

Thus, we can calculate the optimal weights for criteria through the following programming problem (Rezaei, 2015):

$$\min \max_i \left\{ \left|\frac{x_B}{x_i} - v_{Bi}\right|, \left|\frac{x_i}{x_W} - v_{iW}\right| \right\}$$

Subject to

$$\sum_n x_i = 1$$

$$x_i \geq 0 \qquad \forall i = 1, 2, ..., n; \qquad \text{(P1)}$$

Problem (P1) is equivalent to the following linear programming formulation (P2):

$\min \ \phi$

Subject to

$$\begin{aligned}
&\left|x_B - v_{Bi}x_i\right| \le \phi && \forall i = 1, 2, \ldots, n \\
&\left|x_i - v_{iW}x_W\right| \le \phi && \forall i = 1, 2, \ldots, n \\
&\sum_i x_i = 1 && \\
&x_i \ge 0 && \forall i = 1, 2, \ldots, n
\end{aligned} \qquad \text{(P2)}$$

The above problem (P2) is linear in nature and has a unique optimal solution. On solving problem (P2), the value of ϕ^* and optimal weights $\left(x_1^*, x_2^* \ldots, x_n^*\right)$ are determined.

Step 2.3.5: Check the consistency of solution.
The nearer the consistency ratio to zero is, the more compatible the decision makers strategy is. By calculating a consistency ratio, we verify the consistency of the solution:

$$\text{Consistency Ratio} = \frac{\phi^*}{\text{Consistency Index}} \quad .$$

Table 2 is used to get the value of the consistency index (Rezaei, 2015).

Table 2. Consistency index table for BWM

v_{Bi}	1	2	3	4	5	6	7	8	9
Consistency index (max)	0.00	0.44	1.00	1.63	2.30	3.00	3.73	4.47	5.23

The consistency ratio value nearer to' 0' is more consistent, while values nearer to' 1' are less consistent. The consistency ratio not equal to zero (0) means that we don't have full consistency in the pair comparison matrix but multiple optimality. We will, therefore, find the optimal intervals for criteria weights as given by Rezaei (2015). The lower and upper bound of the weights of i^{th} criteria by solving the following problems (P3) and (P4) respectively:

$$\begin{aligned}
&\min \ x_i \\
&\mathit{Subject\ \ to} \\
&\left|\frac{x_B}{x_i} - v_{Bi}\right| \le \phi^* && \forall i = 1, 2, \ldots, n \\
&\left|\frac{x_i}{x_W} - v_{iW}\right| \le \phi^* && \forall i = 1, 2, \ldots, n \\
&\sum_i x_i = 1 && \\
&x_i \ge 0 && \forall i = 1, 2, \ldots, n.
\end{aligned} \qquad \text{(P3)}$$

$$\begin{aligned}
&\max \ x_i \\
&Subject \ \ to \\
&\left| \frac{x_B}{x_i} - v_{Bi} \right| \le \phi^* \qquad \forall i = 1, 2, ..., n \\
&\left| \frac{x_i}{x_W} - v_{iW} \right| \le \phi^* \qquad \forall i = 1, 2, ..., n \\
&\sum_i x_i = 1 \\
&x_i \ge 0 \qquad\qquad \forall i = 1, 2, ..., n.
\end{aligned} \tag{P4}$$

The minimum and maximum values of the weights of the criteria are generated by solving the above programming problems (P3) and (P4) for all the criteria. The center of these intervals gives us the rank the criteria or alternatives.

2.4 Two Way Assessment

In this section, we incorporate the stakeholder opinion into the decision making process by determining the severity values for each of the vulnerability attributes. In this phase, we incorporate the stakeholder's and developer's opinion about the critical nature of each vulnerability attributes, to calculate the utility values in terms of their criticality (Kapur et al., 2014). The stakeholders are asked to prioritize the vulnerabilities based on its level of criticality. In this study we take five values of criticalities of each vulnerability, given by the set $H = (h_1, h_2, ..., h_5)$, which is also termed as acceptance scale. The values of the acceptance scale taken in this study are 2, 4, 6, 8 and 10 where h_1 represents high level of criticality with having a value 10, h_2 signifies that the vulnerability is critical with the value 8, similarly h_3, h_4 and h_5 represents medium, slight and low level of vulnerabilities. The vulnerabilities are defined through the set $V = (v_1, v_2, ..., v_n)$, while the set $X = (x_1, x_2, ..., x_n)$ gives us the weights taken from BWM as obtained from P3 and P4 in step 2.3.5. The responses of the stakeholders are collected in form of pairwise comparisons denoted by $C_{ij,}$ which expressed in terms of percentage based on the response of the stakeholders towards its criticality. For example, if out of twenty (20) stakeholders, ten (10) believe that the first attribute is highly critical, then the value of C_{11} =0.5 (50%) while the remaining 50% stakeholders believe that attribute V1 is critical then C_{12}=0.5. Then, we calculate the expected level weight (E) by multiplying the values of C_{ij} of each attribute with their respective level of acceptance scale (H) and summing across each vulnerability attribute. For assessing the individual perception of each attribute, we multiply the weight of each attribute with its respective expected level weight and sum of all individual utilities give us the overall utility measure of the method as given in Table 3.

Table 3. Overall utility measure for BWM

Vulnerabilities	**BWM weights**	**LEVELS**					**Expected weight level**	**Contribution to Total Expected Utility (Ui)**
		Highly Critical	**Critical**	**Medium Critical**	**Slightly Critical**	**Least Critical**		
		h_1	h_2	h_3	h_4	h_5		
V_1	x_1	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	$E_1 = \sum_j c_{1j} h_j$	$E_1 x_1$
V_2	x_2	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	$E_2 = \sum_j c_{2j} h_j$	$E_2 x_2$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
V_n	x_n	C_{n1}	C_{n2}	C_{n3}	C_{n4}	C_{n5}	$E_n = \sum_j c_{nj} h_j$	$E_n x_n$
Total Utility								$\sum_i E_i x_i$

Graphically the research methodology is presented as shown in Figure 1.

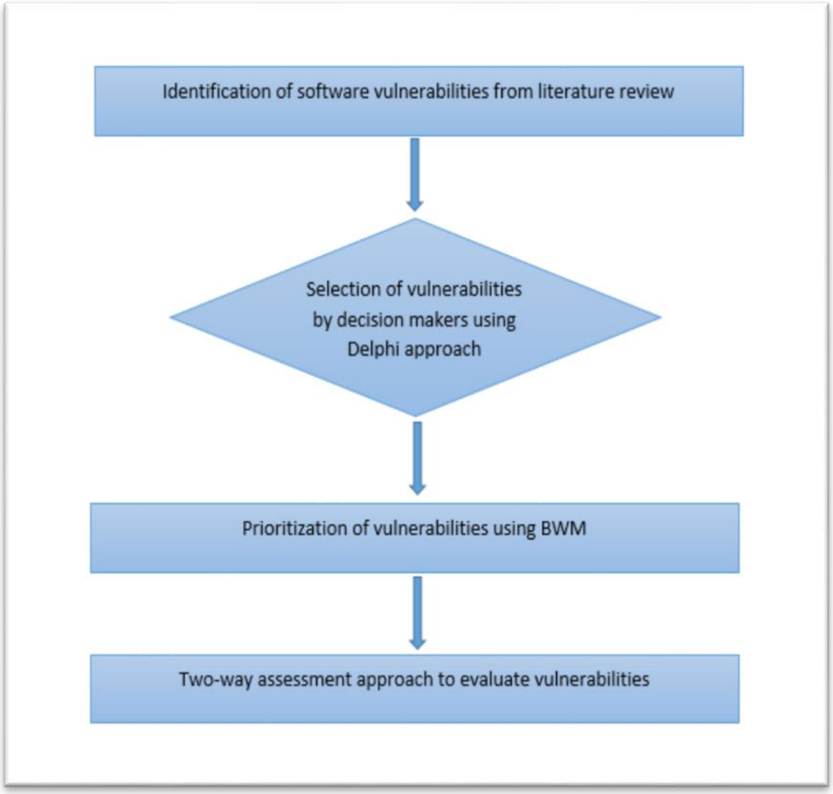
Identification of software vulnerabilities from literature review
Selection of vulnerabilities by decision makers using Delphi approach
Prioritization of vulnerabilities using BWM
Two-way assessment approach to evaluate vulnerabilities

Figure 1: Graphical representation of research methodology

3. Numerical Illustration

As mentioned above, we address the vulnerabilities of the software and its dangerous consequences if not treated correctly. In this section we present the numerical illustration to confirm the proposed framework.

3.1 Prioritizing Vulnerabilities Using BWM

Following the steps of BWM, as given in step 2.3, we have ranked the vulnerabilities based on their criticality. Since BWM requires less number of comparisons, decision makers (DMs), thus, select the best/highly critical and the worst/least critical criteria using step 2.3.1. SV1 is chosen as the highly critical and SV2 as least critical vulnerability by DMs. The steps 2.3.2 and 2.3.3 allow DMs to give their preference of best criterion to others (BO) and others to worst (OW) criteria as shown in Table 4.

Table 4. Best-to-Others (BO) and Others-to-Worst (OW) pair-wise comparison matrix.

Vulnerabilities	BO	OW
SV1	1	9
SV2	5	5
SV3	8	2
SV4	4	6
SV5	7	3
SV6	6	4
SV7	2	8
SV8	3	7
SV9	9	1

Following step 2.3.4 we use the linear programming problem P2 to discover the appropriate weights. On solving P2, the ideal weight value and ϕ^* is calculated. The value of ϕ^* which is equal to 1.72580 is used to check the consistency of the solution as given in step 2.3.5. The CR = 1.72580 / 5.23 yields to the solution 0.33 which is closer to 1, means consistency is less. Hence, we may have multiple optimality in our solution. Now using the programing problems (P3) and (P4) of section 2.3.5, the maximum and minimum values of the weights of the criteria are generated and then the center of these intervals (average) can be used to rank the criteria as shown in Table 5.

Table 5. Weights of the vulnerabilities calculated

Vulnerability	Min	Max	Centre	Normalized Weight	Rank
SV1	0.243	0.302	0.273	0.270	1
SV2	0.074	0.092	0.083	0.082	5
SV3	0.025	0.047	0.036	0.036	8
SV4	0.098	0.131	0.115	0.113	4
SV5	0.029	0.056	0.043	0.042	7
SV6	0.052	0.070	0.061	0.060	6
SV7	0.154	0.250	0.202	0.200	2
SV8	0.129	0.218	0.173	0.171	3
SV9	0.023	0.028	0.025	0.025	9

3.2. Two-Way Assessment Technique

The severity of each criterion is calculated using two-way assessment approach. In the present study, we have applied two-way assessment technique in-order to find the overall criticality of selected vulnerabilities. The vulnerabilities ranked by the DM are based on the level weights of set (H). For example, the weight calculated from BWM for SQLI is 26.961 and 80% of the respondents rank V1 as highly critical and remaining 20%, rank it as critical. To obtain the expected level weight, we multiply (10*0.8) +(8*0.2) + (6*0) + (4*0) + (2*0) which is equal to 9.6. Also by multiplying weights calculated from BWM with expected level weights i.e., 26.961 * 9.6= 258.828 gives us the individual criticality of SQLI. In similar manner, we calculate the overall criticality of all the vulnerabilities represented in the Table 6.

Table 6. Overall criticality measure of vulnerabilities

Vulnerabilities	**Weights**	**LEVELS**					**Expected Level Weight**	**Contribution to Total Expected criticality (U_i)**
		Highly Critical	**Critical**	**Medium Critical**	**Slightly Critical**	**Least Critical**		
		10	**8**	**6**	**4**	**2**		
SV1	26.961	0.8	0.2	0	0	0	9.6	258.828
SV2	8.233	0	0	0.6	0.4	0	5.2	42.810
SV3	3.594	0	0	0.4	0.4	0.2	4.4	15.815
SV4	11.327	0	0	0.2	0.6	0.2	4	45.307
SV5	4.227	0	0.2	0.4	0.4	0	5.6	23.672
SV6	6.035	0	0	0.4	0.6	0	4.8	28.970
SV7	19.977	0.6	0.4	0	0	0	9.2	183.787
SV8	17.132	0.4	0.4	0.2	0	0	8.4	143.908
SV9	2.514	0	0	0	0.2	0.8	2.4	6.033
Total criticality								**749.129**

4. Results and Discussions

The present study aims in understanding the prioritization of vulnerabilities so that the developers and testers know their fixing order. We have identified nine vulnerabilities based on inputs from the experts. In this paper, a new MCDM technique known as BWM was utilized to prioritize the vulnerabilities. Further, in order to assess the vulnerabilities in terms of their utility, we have applied two-way technique. The two-way assessment helps in considering the critical nature of the vulnerability. In this technique the stakeholders are involved, who in addition to the weights given by the expert panel, rates the vulnerabilities on the level of their criticality. The more the weight of the utility value the more is its critical level of vulnerability, and thus, needs to be resolved quickly. The ranking is done using BWM, it is visible from Table 5 that vulnerability SV1 has maximum weightage value of 0.270 and is on Ist rank. SV7 is placed on 2nd rank with weightage value 0.200 and SV8 with weightage value of 0.171 has been placed on rank 3rd. Ranking of remaining vulnerabilities in their descending order along with their weights are: SV4 (0.113) > SV2 (0.082) > SV6 (0.060) > SV5 (0.042) > SV3 (0.036) > SV9 (0.025).

The Figure 2 represents the weights of the vulnerabilities when we solve problem P3 and P4. In the final results of the BWM, we have used the average of the minimum and maximum values. It can be seen from the Figure 2 below, that vulnerabilities SV1, SV7 and SV8 are highly critical while vulnerability SV9 is having very low criticality. It can also be observed that the difference between the maximum and minimum values for SV7 and SV8 is higher as compared to the other values. On the other hand, the minimum and maximum value for SV1 is almost same. This means that interval weights are not simple to classify visually, as some overlap. To prioritize the total importance of vulnerabilities, we, therefore, use a two-way evaluation to consider the stakeholder perspective in the decision-making process.

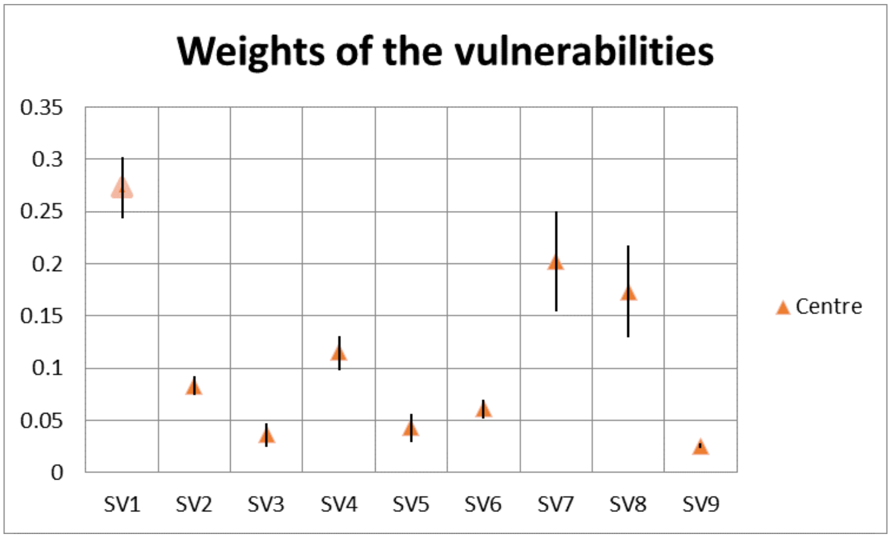

Figure 2. Weights of vulnerabilities using BWM

The total criticality value comes out to be 749.129 along with the individual criticality of each vulnerability in the descending order as SV1(258.828) > SV7(183.787) > SV8 (143.908) > SV4(45.307) > SV2(42.810) > SV6(28.970) > SV5(23.672) > SV3(15.815) > SV9(6.033). It can be seen that the maximum utility is from the SV1, SV7 and SV8. Thus, it can be concluded that maximum efforts must be put to resolve these vulnerabilities. Once we obtain the total criticality value, we calculate the ideal best, ideal worst and threshold scenarios. Ideal best scenario can be calculated by assuming that the DMs rank all the vulnerabilities as highly critical, thereby giving the ideal best value (Ui) = 1000. The ideal worst scenario can be calculated by taking into consideration that each DM marks all the vulnerabilities as least critical, thereby resulting in the ideal worst value (Ui) = 200. Also the threshold value is calculated by considering that DMs have marked all the vulnerabilities as medium critical and thus results in the value (Ui) = 600. The results calculated from Table 5 lead us to the conclusion that our total criticality value (Ui) = 749.129 is more than the threshold value and closer to ideal best as well and so the total criticality value is acceptable. Graphically the ideal worst, ideal best and optimal criticality values are shown in Figure 3.

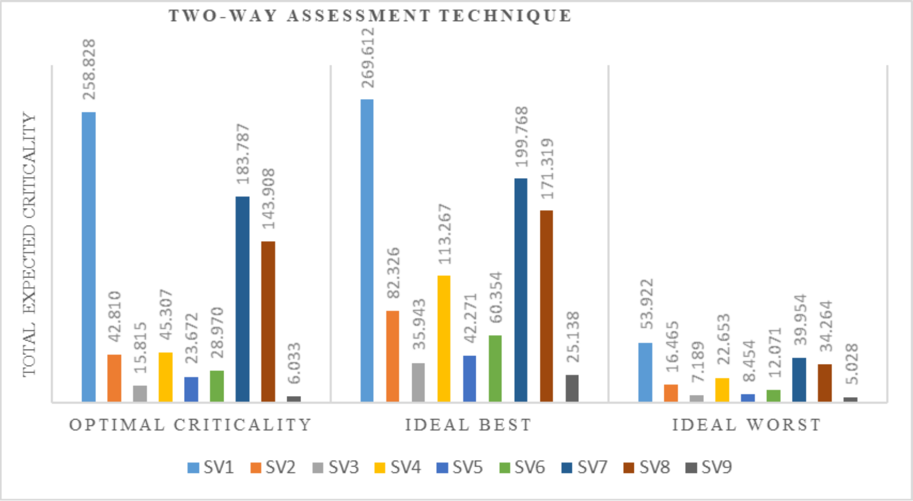

Figure 3. Graphical representation of ideal worst, ideal best and optimal values of criticality

While calculating the ideal worst and ideal best scenarios, it is quite visible that the ranking of vulnerabilities remains the same in all the three scenarios. For example, vulnerability SV1 is having the maximum criticality and SV9 is having the lowest value in all the three scenarios. However, there lies a huge difference in the individual as well as total criticality values among ideal best and ideal worst scenarios. The vulnerability SV1 is having the individual criticality value 269.612 in ideal best scenario while as, in ideal worst the same vulnerability is having the individual criticality score equal to 53.922 and in optimal case the value is 258.828. likewise, the individual criticality values of all the vulnerabilities obtained by two-way assessment approach are closer to their ideal best values. Thus, we can say that the results obtained in terms of optimal criticality values of the vulnerabilities are acceptable and need to be forwarded to security team for immediate actions accordingly.

5. Conclusion

The present study deals with the prioritization of software vulnerabilities. The increase in the dependence computer systems has made software security a crucial aspect. The foremost step for any developer and tester is to understand the critical nature of the vulnerability, so that vulnerabilities which can harm the system badly can be resolved first. The literature mostly considers the modeling aspect of vulnerabilities with little focus on prioritization aspect. The present paper aims in overcoming this literature gap by prioritizing the software vulnerabilities based on their criticality. The inputs of both the developers and the stakeholders have been considered for the prioritization process.

An integrated two-fold approach has been proposed in the present study for prioritizing the software vulnerabilities. A total of nine (9) vulnerabilities are identified from the literature. In the first phase, BWM is used for prioritize the vulnerabilities, the result of which are used in the second phase for finding the overall criticality of vulnerabilities using two-way assessment. The results thus obtained can be utilized by the testing group to plan their testing process more efficiently so as to minimize the cause and effect of vulnerabilities.

The study can be further explored by considering additional vulnerability attributes. Since we have utilized BWM and two-way approach in our paper, other MCDM techniques can also be applied in future taking into account the uncertainty of data as well. Further, we can apply mathematical modelling to improvise the findings.

Conflict of Interest
The authors confirm that there is no conflict of interest for this publication.

Acknowledgements
The authors would like to express their sincere thanks to the editor and anonymous reviewers for their time and valuable suggestions.

References

Arora, A., Krishnan, R., Telang, R., & Yang, Y. (2010). An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure. *Information Systems Research*, *21*(1), 115-132.

Govindan, K., Jha, P.C., Agarwal, V., & Darbari, J.D. (2019). Environmental management partner selection for reverse supply chain collaboration: A sustainable approach. *Journal of Environmental Management*, *236*, 784-797.

Huang, C.C., Lin, F.Y., Lin, F.Y.S., & Sun, Y.S. (2013). A novel approach to evaluate software vulnerability prioritization. *Journal of Systems and Software*, *86*(11), 2822-2840.

Jimenez, W., Mammar, A., & Cavalli, A. (2009). Software vulnerabilities, prevention and detection methods: a review1. *Security in Model-Driven Architecture*, *6*, 1-56.

Kansal, Y., Kapur, P.K., Kumar, U., & Kumar, D. (2017). User-dependent vulnerability discovery model and its interdisciplinary nature. *Life Cycle Reliability and Safety Engineering*, *6*(1), 23-29.

Kapur, P.K., Nagpal, S., Khatri, S.K., & Yadavalli, V.S. (2014). Critical success factor utility based tool for ERP health assessment: a general framework. *International Journal of System Assurance Engineering and Management, 5*(2), 133-148.

Kapur, P.K., Pham, H., Gupta, A., & Jha, P. (2011). *Software reliability assessment with OR applications*. Springer, London.

Kapur, P.K., Singh, G., Sachdeva, N., & Tickoo, A. (2014, October). Measuring software testing efficiency using two-way assessment technique. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization.* (pp. 1-6). IEEE. Noida, India.

Khurana, D.K., Kapur, P.K., & Sachdeva, N. (2017). Utility based tool to assess overall effectiveness of HRD instruments. *International Journal of Business Analytics*, *4*(2), 20-36.

Liu, Q., & Zhang, Y. (2011). VRSS: A new system for rating and scoring vulnerabilities. *Computer Communications*, *34*(3), 264-273.

Liu, Q., Zhang, Y., Kong, Y., & Wu, Q. (2012). Improving VRSS-based vulnerability prioritization using analytic hierarchy process. *Journal of Systems and Software*, *85*(8), 1699-1708.

Lyu, M.R. (1996). *Handbook of software reliability engineering*. IEEE Computer Society Press, Hightstown, NJ, USA.

Narang, S., Kapur, P.K., & Damodaran, D. (2017, December). Severity measure of issues creating vulnerabilities in websites using two way assessment technique. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS)* (pp. 309-316). IEEE. Dubai, United Arab Emirates

Okoli, C., & Pawlowski, S.D. (2004). The Delphi method as a research tool: an example, design considerations and applications. *Information & Management*, *42*(1), 15-29.

Ozkan, S. (1999), CVE details, the ultimate security vulnerability data source. Technical report. Retrieved from http://www.cvedetails.com. Accessed on 2 Feb 2019.

Rezaei, J. (2015). Best-worst multi-criteria decision-making method. *Omega*, *53*, 49-57.

Rezaei, J. (2016). Best-worst multi-criteria decision-making method: some properties and a linear model. *Omega*, *64*, 126-130.

Sharma, R., Sibal, R., & Sabharwal, S. (2019). Software vulnerability prioritization: a comparative study using TOPSIS and VIKOR techniques. In *System Performance and Management Analytics*. Springer, Singapore, pp. 405-418.

Sibal, R., Sharma, R., & Sabharwal, S. (2017). Prioritizing software vulnerability types using multi-criteria decision-making techniques. *Life Cycle Reliability and Safety Engineering*, *6*(1), 57-67.

Links and Websites

CERT. Vulnerability Notes and description. Retrieved from http://www.kb.cert.org/vuls/html/fieldhelp#metric.

CVSS. (2007). Common Vulnerability Scoring System. Retrieved from http://www.first.org/cvss/history

Google. Severity Guidelines for Security Issues. Retrieved from http://dev.chromium.org/developers/severity-guidelines

Microsoft. (2002). Microsoft Security Responce Center. Retrieved from http://www.microsoft.com/technet/security/bulletin/rating.mspx.

Mozilla. Mozilla foundation security Advisories. Retrieved from http://www.mozilla.org/security/announce/.

National Vulnerability Database, N. Retrieved from http://nvd.nist.gov/.

Red-hat. Classification of security Issues. Retrieved from http://www.redhat.com/f/pdf/rhel4/SecurityClassification.pdf

Secunia. Secunia research. Retrieved from http://secunia.com/about_secunia_advisories/

Symantec Corporation, U. S. (2000). Retrieved from https://www.symantec.com/security-center

Symantec Corporation [US], (2018). Internet Security Threat Report. Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-executive-summary-en.pdf

Symantec Corporation [US], (2013). *Internet Security Threat Report*. Retrieved from https://www.symantec.com/content/en/us/enterprise/other_resources/bistr_main_report_v18_2012_21291018.en-us.pdf

VUPEN (2005). Vulnerability Research and Intelligence. Retrieved from http://www.vupen.com/.

X-Force, 1999. X-Force frequently asked questions, Available at: http://www-935.ibm.com/services/us/iss/xforce/faqs.html.