Received: 20 July 2025 | Revised: 20 October 2025 | Accepted: 31 October 2025 | Published online: 26 December 2025

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 00(00) 1–5
DOI: 10.47852/bonviewJCCE52026871

BON VIEW PUBLISHING

Measuring Satisfaction of Security Requirements Using Fuzzy Soft Set Theory

Dyala Ibrahim[1,*], Omar Isam Al Mrayat[2], Lial Raja Alzabin[3], Ahmad Alajarmeh[4], Ahmad Reda Alzighaibi[5], Hasan Hashim[5], El-Sayed Atlam[6,7] and Malik Jawarneh[3]

[1] *Department of Cyber Security, Amman Arab University, Jordan*

[2] *Department of Software Engineering, Amman Arab University, Jordan*

[3] *Department of Computer Science, Amman Arab University, Jordan*

[4] *Department of Data Science and Artificial Intelligence, Amman Arab University, Jordan*

[5] *Department of Information Systems, Taibah University, Saudi Arabia*

[6] *Department of Computer Science, Taibah University, Saudi Arabia*

[7] *Department of Computer Science, Tanta University, Egypt*

Abstract: In recent years, a range of innovative approaches have been developed to tackle decision-making challenges more effectively. These approaches include concepts such as probability theory, fuzzy set theory, intuitionistic fuzzy sets, vague sets, interval mathematics, and rough set theory. These tools each offer unique approaches to managing uncertainty and imprecision, which are common when working with complex systems. One major issue in software development is failure to meet security requirements. This can be caused by a variety of factors, such as unclear security specifications, evolving threat landscapes, or gaps in communication among stakeholders. To address this persistent problem, this paper introduces a new method designed to quantify security requirements more accurately and streamline the decision-making process for assessing security risks. This approach is based on fuzzy soft set theory, which allows for more flexible and precise handling of uncertain or ambiguous information. By using this method, software engineers and security professionals can gain a clearer understanding of security risks and make better-informed decisions when building secure systems. To demonstrate the practical value of the proposed method, a case study is conducted to evaluate its effectiveness. The results of this case study suggest that security administrators can use this method to implement more targeted and efficient security measures, ensuring that the systems they develop are more resilient to potential threats.

Keywords: security requirements, software development, software engineers, software systems, software vulnerabilities

1. Introduction

Software systems often face quite critical security threats [1]. Over the years, technological advances have had both negative and positive impacts. Systems [2] have become easier for intruders to attack. Therefore, we believe it essential to emphasize requirements elicitation to prevent intentional and unintentional attacks. It is crucial to consider security issues along with other system elements from the outset. Security is a non-functional requirement (NFR) that is extremely essential and unique in its requirements; however, it must be associated with all other functional and NFRs and organized into productive architectures, models, and applications [3].

Developers must identify vulnerabilities along with their impact on the overall system and each security requirement as part of the Secure Software Development Life Cycle (SSDLC). They should also understand the security needs to mitigate relevant software vulnerabilities. Determining software vulnerabilities is crucial because not addressing vulnerabilities at the beginning stage of the Software Development Life Cycle (SDLC) will incur a sharp increase in costs in the later stages [4, 5]. Evaluating the security posture of software remains a challenge, specifically determining whether adequate security measures have been implemented throughout the SDLC [6, 7]. The proposed method aims to address these issues and make a meaningful contribution to this field of study.

Software security encompasses three key objectives: Confidentiality, Integrity, and Availability of information resources created, processed, and stored by any software [8]. Similarly, confidentiality means avoiding unauthorized disclosure of critical data. In contrast, integrity refers to protecting information from illicit modifications, while availability refers to preventing unauthorized damage, access, or denial of service [9]. Unfortunately, developers often fail to prioritize security from the very start of software development. Instead, security is usually only considered after the software is fully developed [10]; consequently, most software is vulnerable to malicious attacks [11]. Based on the aforementioned circumstances, there is an urgent need for security in the software development process.

The Security Requirements Engineering (SRE) process is one of the most disregarded factors in the SDLC [12]. One of the most

***Corresponding author:** Dyala Ibrahim, Department of Cyber Security, Amman Arab University, Jordan. Email: d.ibrahim@aau.edu.jo

critical factors contributing to this oversight is that conventional requirements engineering methods often fail to meet security needs [13]. The requirements engineering fails to effectively meet the security needs because of two reasons: First, security requirements are often overlooked during the necessary investigation and modeling process; Second, developers lack the competence to develop secured software. Regarding security, according to [14], the concept behind requirements engineering is one of the most easily overlooked parts of improving the software development life cycle, which is the security requirements engineering process [15, 16]. This paper proposes a new method to minimize attacks that negatively affect developed software.

In this paper, we apply fuzzy soft set theory to assess security requirements in software applications. According to existing research, it is highly effective in decision-making. The main objective of this paper is to improve current methods by identifying potential security factors, such as vulnerabilities and errors, to ensure that all the security needs of a business are met. This approach will enable developers to better assess and quantify the security requirements of the systems they build. Furthermore, it can help developers allocate resources more effectively to strengthen software security and reduce vulnerabilities, and the proposed method is evaluated through a case study. A study was conducted to quantify security requirements and develop efficient, accurate decision-making for assessing security risks in the software industry. The results were highly accurate in that the priorities for security requirements when developing software are in complete agreement with the standards of the National Institute of Standards and Technology (NIST) and the institute for SysAdmin, Audit, Networking, and Security (SANS Institute).

The motivation for this work is to contribute to the field of secure software engineering, especially in terms of security requirements, by prioritizing the security requirements of each functional requirement and assigning weights to each security requirement based on the implementation results of the fuzzy soft set algorithm. The main problem in recent papers is that authors focus only on the security requirements in the implementation phase; however, this is a significant problem if they are not considered in the early stages of the SDLC, particularly the requirements and design phases. This will lead to system errors and vulnerabilities, because in the software industry, security requirements must be considered to ensure secure software.

The paper is organized as follows: Section 2 discusses related works, Section 3 discusses the preliminary for this paper, and Section 4 presents the proposed method. Section 5 highlights the testing and validation of the efficacy and accuracy of the proposed method. Section 6 presents the conclusion.

2. Related Works

This section will discuss recent approaches to decision-making in uncertain or imprecise environments, which have become increasingly important. The available literature indicates that fuzzy soft sets may be a good solution for decision-making problems [17, 18]. Feng et al. [19] proposed a decision-making framework based on the fusion of fuzzy set and soft set theory to manage uncertainty in decision-making. By introducing adjustable parameters, this method enables decision-making to be tailored to each specific problem, offering greater flexibility and accuracy in ambiguous situations. This approach takes into account partial truths and contradictions. Groups and individuals must be assigned values that correspond to probabilities. This provides a better context for decision-making. Hence, it applies to real problems. Moreover, other methods may also fail to find solutions. Study results show that the adjustable fuzzy soft set model provides better decision-making support in complex situations. This presents opportunities for future research output. On the other hand, Alcantud [20], Hussain et al. [21], and Mahmood [22] studied the problem of uncertainty-based decision-making with fuzzy soft sets from data inputs of multiple observers. The authors propose an algorithm that aims to improve existing methods by enhancing the fusion of information and decision-making.

Furthermore, Alcantud [20] created a new algorithm that can make better decisions under uncertain conditions and from information provided by different agents. The method combines fuzzy sets (which handle uncertainty) and soft sets (which organize data based on parameters), enabling it to bring together different ideas more effectively. The study showed that this approach works well even with incomplete or subjective data, making it more dependable and flexible than earlier methods for complex, real-world decision-making [21].

Moreover, Hnaini et al. [10] proposed a new method for specifying security requirements. The proposed security requirement template is processed into two cycles to tackle the problem of security requirements. The proposed template has been implemented and identified, and imitations have been resolved by correctly defining security requirements. The proposed template can accurately and correctly identify security requirements. Furthermore, Selvaraj and Vadivel [23] proposed a new framework for solving the security requirements problem. The results prove the efficiency of the proposed framework against security risks.

Moreover, Nhlabatsi et al. [24] proposed a new model to tackle the problems of quantifying an intrusion system's security attributes. The proposed model places the attacker randomly in the model. So randomness facilitates capturing the attacker's behavior and the system's response to security intrusions. The proposed model has been evaluated using statistical metrics such as the mean time or effort required for a security failure and the probability of a security failure based on errors in security attributes. Their work got satisfactory results.

Furthermore, Andrade et al. [25] proposed an exploratory study to address the problem of quantifying security requirements through the initial stages of the development process. Their focal objective is to introduce an integrated security model by software developers in the elicitation processes, facilitated through the use of abuse stories. This model inherently contains critical elements such as modern technologies, human factors, and security management for formulating abuse stories and their integration within agile methodological processes.

Al-Matouq et al. [26] proposed a new secure software design maturity model (SSDMM). The experimental analysis and results showed that the SSDMM provided a new measure for the maturity level of software development organizations.

Tešić et al. [27] proposed an improved fuzzy-based multi-criteria decision-making (MCDM). The model proposed examines how fuzzy theory helps to quantify qualitative data to simulate real-life situations. The Defining Interrelationships Between Ranked (DIBR II) technique is an innovative method of dealing with uncertainties and incomplete information in MCDM. The DIBR II method uses triangular fuzzy numbers with adjustable confidence intervals. The results proved the effectiveness of their work.

On the other hand, Alojaiman [28] focused on the country's shift to a new and improved way of advancing digital transformations. Using ICTs and modern communications infrastructure achieves this goal. Furthermore, the study proposed a system based on an MCDM technique, specifically a fuzzy Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), for the efficient selection of Internet of Things (IoT) applications. An engineering prototype and a hierarchy were built to evaluate and compare a set of important criteria based on expert preferences and suggestions. Analysis has shown that a T5 IoT application was selected as the best and most reliable by both fuzzy TOPSIS and TOPSIS.

In addition, Niazi et al. [29] modified Somerville's requirement engineering practices. The security parameter was considered through the best practices process. After identifying best practices and a framework for secure requirement engineering, the Requirements Engineering Security Maturity Model (RESMM) was devised.

Furthermore, Muhammad et al. [30] presented an updated method to identify and prioritize NFRs, which can be challenging given that this stakeholder may be unknowledgeable about NFRs. Most stakeholders would only be familiar with quality attributes such as performance, efficiency, and security. However, there are many other quality attributes, including confidentiality, scalability, usability, maintainability, portability, and reliability. The stakeholders may not be aware of these attributes. Ignoring important NFRs and incorporating irrelevant ones into the final product will lead to a decline in the quality of the final product. Moreover, it can take considerable time to gather and prioritize NFRs from relevant stakeholders. This is particularly a challenge in agile software development environments where products are developed and delivered rapidly. To address these issues, this study proposed a model for prioritizing NFRs in agile software development to ensure the production of high-quality software. The results showed that this prioritization model helps reduce the time spent on meetings and discussions required to identify common NFRs in agile projects.

3. Preliminary

This section will discuss the fuzzy soft set theory approach used in this proposed method. The fuzzy soft set theory will be used to quantify and prioritize the security requirements in software application development because it is amazingly effective in decision-making [31]. It represents a potential security factor (vulnerabilities and errors) that encompasses all business needs related to security. This contribution helps developers quantify, prioritize, and measure the security requirements in the proposed systems. Moreover, it can help software developers allocate more resources to improve security in any software and reduce vulnerabilities [32, 33].

3.1. Fuzzy soft set theory

A fuzzy soft set is a collection of items where each item has a degree of membership to the set, ranging from complete membership to non-membership [34]. A membership function determines this degree. The standard set operations, such as union, intersection, and complement, can be adapted for fuzzy sets [35]. This study needs an accurate mathematical decision-making tool, so we chose fuzzy soft set theory. The Cagman, Citak, and Enginoglu Algorithm (CCEA) [36] will be used in constructing a more efficient decision process and calculating the Security Index (SI). Now, we recall the definition of fuzzy decision set and the CCEA given in the paper.

Definition: Let $F_X \in$ FPS (U). Then the fuzzy decision set of F_X, denoted as F_{dx}, is defined as

$$F^d{}_X = \{\mu_{F^dX}(u)/u : u \in U\} \tag{1}$$

where $F^d{}_X$ is a function set for the fuzzy decision set of U
$\mu F^d{}_X$ is the decision of $F^d{}_X$ (membership function)

When a fuzzy set is defined on U, its membership function $\mu F^d{}_X$ is defined as

$$\mu F^d{}_X : u \to [0,1] \tag{2}$$

$$\mu F^d{}_X = \frac{1}{|\text{supp}(X)|} \sum_{x \in \text{supp}(X)} \mu_X(x) \chi_{fX(x)}{}^{(u)} \tag{3}$$

where supp(x) is the support set of X
(X) is the crisp subset, which is determined by the parameter x and characteristic function (see Equation (4)).

$$Xf_X(x)^{(u)} = \begin{cases} 1, & if\, u \in f_X(x), \\ 0, & if\, u \notin f_X(x), \end{cases} \tag{4}$$

Where $f_x(X)$ belongs to the n-th element of the universe U, it will be assigned a value of 1; otherwise, if $f_x(X)$ does not belong to the n-th element of U, it will be assigned a value of 0. Then, we use the vulnerability I (Vi) based on the Security Index (SI) in Equation (5).

$$SI^{Vi} = \sum_{i,j \in I} V_i E_j / j \tag{5}$$

SI^{Vi} is a security index for vulnerability i.

Once we have the fuzzy decision set of the fuzzy soft set, we may need to choose the best single alternative. Therefore, we can make the following decisions.

Step 1: Construct a soft set Fx over U.
Step 2: Compute the fuzzy decision set F^dx.
Step 3: Select the largest membership grade μF^dx.

Next, we introduced the proposed method through a case study to show the results of using fuzzy soft set theory. In normal use, the CCEA calculates μF^dx once for the parameters to obtain the decision for universal (U). For instance, a customer wants to buy a car that meets the following parameters (E) (expensive, inexpensive, made in the USA or Japan, number of seats, color, and model year 2024). The parameters that the customer prioritizes are (inexpensive, made in the USA, number of seats, color, and model year 2024), respectively. As per the above parameters, the experts will make a decision based on the customer's requirements while taking into account the customer's budget.

3.2. Standard soft set operations

The three-way decision theory offers a useful approach for making decisions when there is uncertainty or incomplete information, especially when a simple yes-or-no decision is not enough. Soft sets help to model and represent specific types of uncertainty in these situations [37]. This section presents a quick overview of the fundamental concepts of soft sets and presents two different interpretations of soft sets [38].

Suppose (S, E) is a soft set on a nonempty set U. Alcantud et al. [25] defined the core and the support concepts of the soft set (S, E) as follows:

$$\begin{aligned} &\text{Core}\ (\mathrm{S},\mathrm{E}) = \cap_{\mathrm{e}\in\mathrm{E}}\mathrm{S}(\mathrm{e}) \\ &\text{Support}\ (\mathrm{S},\mathrm{E}) = \cup_{\mathrm{e}\in\mathrm{E}}\mathrm{S}(\mathrm{e}) \ldots\ldots = \{\mathrm{x}\ \in \cup\, |\forall_{\mathrm{e}} \in\ \mathrm{E}, \mathrm{x} \in \\ &\mathrm{S}(\mathrm{e})\}, = \{\mathrm{x}\ \in \cup |\exists_{\mathrm{e}} \in \mathrm{E}, \mathrm{x} \in \mathrm{S}(\mathrm{e})\} \end{aligned} \tag{6}$$

4. Proposed Method

In this section, the proposed method is presented as follows:

1) Calculate $\mu F^d{}_X$ of the parameters (E) for each element inside (U) to obtain the decisions for the universal vulnerabilities (U) according to (E) values. For instance, a security expert might want to quantify all security requirements in his proposed system using the following parameters (E): occurrence one time, occurrence two times, high damage, low damage, and impact on critical assets. According to the SANS Institute, as supported by the NIST, the (U) consists of three vulnerabilities, which are an SQL injection. As mentioned above, the security expert will apply the algorithm three times to obtain the vulnerabilities index (V_I) based on the (E) values.
2) Construct a universe (U) containing a set of elements (vulnerabilities or errors) to which we base our decision-making process. Construct the power of the universe (E_u), a set of parameters describing all aspects of the elements in (U). Finally, give fuzzy values (E) to all parameters (power of the universe [E_u]).

3) Compute the fuzzy set on (U), which is a characteristic function (see characteristic function in Equation (4)).
4) Find F ($e_{u,n}$) for all parameters, i.e., construct a set for each parameter containing elements with this aspect (parameter) (eu,1), (eu,2),…($e_{u,n}$). Compute the fuzzy decision using this mathematical algorithm related to the work by Madan et al. [1] (see Equation (3)).
5) When the algorithm is applied, if this element (vulnerability or error) is available in F ($e_{u,n}$), we substitute it with 1 to multiply it with ($e_{u,n}$) value, i.e., if U∈ F ($e_{u,n}$), then $Fdx = 1 \times (e_{u,n})$ value; otherwise, we substitute with 0 to multiply it with ($e_{u,n}$) value, i.e., if U∉ F ($e_{u,n}$), then $Fdx = 0 \times (e_{u,n})$ value, as shown in the characteristic function (see Equation (4)). As defined, after applying the algorithm to all elements, we obtain a fuzzy value for all elements. This value describes the element's situation based on the parameters used. For example, if the fuzzy value is near 1, the element with that value has high priority; if it is near 0, it indicates low priority. Now, we use Equation (3) to solve the problem in the following case study.

and error indices will be calculated, resulting in an input used to find the SI. Second, merging the security requirements with the SI is required to obtain the quantification of security requirements (security requirements index). Table 1 shows the vulnerabilities, errors, and security indices with an example for each type. The case study is a crucial part of our research, and we chose Modanisa[1] as our example. It is an ideal choice because it is a popular e-commerce site that manages sensitive information—such as customer data, financial transactions, and user logins—making it perfect for analyzing real-world security needs.

5. Research Experimental Analysis and Results

The following sections will discuss the primary metrics used to evaluate the proposed model.

Table 1
Demonstration for vulnerabilities, error index, and security index with an example for each definition

Parameter Type	Vulnerability Index	Error Index	Security Index
Definition	An index for weakness in a product that could allow an attacker to compromise the integrity, availability, or confidentiality of that product	An index for an undesirable deviation from requirement or to change the functionality of the program	An index for the state of feeling safe, stable, and free from vulnerabilities resulting from errors
Example	Denial-of-service attacks that allow an attacker to cause a Blue Screen of Death	Null pointer references or failed lock releases, rather than redundancy checking	Authorization

This proposed methodology is basically a smart approach to risk ranking that combines facts and judgments—rather than just relying on a checklist. Our goal is to discover the riskiest weaknesses in the software. To achieve this, we use a method called Fuzzy Decision Set. This method helps us estimate the risk associated with every weakness that we discover, such as SQL Injection. This lets us generate a single, clear score, the SI for each weakness, telling us exactly which ones to prioritize fixing.

The Process in Three Simple Steps:

1) Define and weight the criteria: The first step is to sit down with security experts and define the most important risk factors (our "parameters," Eu). These factors might include things such as "how often the vulnerability occurs," "how much damage it causes," or "what kind of crucial assets it affects." The experts then assign a weighted score (a fuzzy value, such as 0.5 or 1.0) to each factor, reflecting its overall importance. For example, "High Damage" always gets a higher score than "Low Damage."
2) Match weaknesses to criteria: Next, we go through every vulnerability (U) and determine which weighted risk factors it satisfies. If a vulnerability is known to cause "High Damage," it gets matched to that parameter.
3) Calculate the final index: The final SI is calculated by adding the scores (weights) of all the criteria a vulnerability satisfies. If a vulnerability satisfies three high-priority criteria, its score will be high. If it satisfies only one low-priority criterion, its score will be low.

In short, the resulting SI value (near 1 for elevated risk, near 0 for low risk) becomes the clear, mathematically justified priority list for the entire security team.

4.1. Case study

The social network Modanisa[1] has been nominated for the case study to quantify security requirements. First, both vulnerabilities

5.1. Vulnerabilities index

Every software has many vulnerabilities, which are considered weak points that allow an attacker to reduce a system's information assurance or deny service, such as missing authentication or authorization [38]. In this section, the vulnerabilities index is calculated according to many parameters:

1) The percentage of vulnerability occurrences in each software, i.e., every vulnerability may happen once or more in a particular software (the likelihood of each vulnerability being exploited).
2) Percentage of the damage that can happen because of this vulnerability to the software.
3) The percentage of the asset is essential in specific software. Each software has its assets, which have different values; each asset may be affected by at least one of the system vulnerabilities. The security experts aim to identify the impact of each vulnerability within the system. Define a universe U as follows: $U = \{V^1, V^2, V^3, V^4, V^5\}$, where V refers to vulnerabilities.

After interviewing the website developer of Modanisa[1] and investigating all vulnerabilities related to this type of system according to SANS and NIST lists [39, 40], the vulnerabilities are

1) SQL Injection (SQLI)
2) OS Command Injection (OSCI)
3) Classic Buffer Overflow (CLBO)
4) Cross-Site Scripting (CR-sS)
5) Download the Code Without Integrity Check (DOCWIC)
6) U= {SQLI, OSCI, CLBO, CR-sS, DOCWIC}

where $E_u = \{e_u\}$ is a set of decision parameters which are related to the U "vulnerabilities" which is specified by the security experts.

$$E_u = \{e_{u,1}, e_{u,2}, e_{u,3}, e_{u,4}, e_{u,5,..} e_{u,n}\}. \quad (7)$$

[1] www.modanisa.com

where E_u= {$e_{u,1}$= occurrence one time, $e_{u,2}$= occurrence two times, $e_{u,3}$= high damage, $e_{u,4}$= low damage, $e_{u,5}$= the influence on very important asset, $e_{u,6}$= the influence on important asset, $e_{u,7}$= the influence on medium asset, $e_{u,8}$= the influence on less important asset}.

In the case study, there are eight parameters (e_u). However, in other cases, one may use more or less than this number of parameters and different values according to the system's nature and significance of the system. The security experts suggest these parameter values based on their importance. For instance, if a vulnerability X occurred two times in the system, it should have high value; see $e_{u,2}$ and vice versa.

$$E = \left\{\frac{e_{u,1}}{0.5}, \frac{e_{u,2}}{1}, \frac{e_{u,3}}{1}, \frac{e_{u,4}}{0.5}, \frac{e_{u,5}}{1}, \frac{e_{u,6}}{0.7}, \frac{e_{u,7}}{0.5}, \frac{e_{u,8}}{0.2}\right\} \quad (8)$$

Each parameter has its equation, i.e., F ($e_{u,1}$) means the parameter ($e_{u,1}$), which is "occurrence one time" and has (0.5) value. All vulnerabilities that occur once will be part of these equations, such as (SQLI, CLBO) vulnerabilities and so on, for all parameters.

F ($e_{u,1}$) = F (Occurrence One Time)
= {SQLI, CLBO}
F ($e_{u,2}$) = F (Occurrence Two Times)
= {SQLI, OSCI, CR-sS}
F ($e_{u,3}$) = F (High Damage)
= {SQLI, CR-sS, DOCWIC}
F ($e_{u,4}$)= F (Low Damage)
= {OSCI, CLBO}
F ($e_{u,5}$)= F (The influence on Very Important Asset)
= {SQLI}
F ($e_{u,6}$)= F (The influence on Important Asset)
= {OSCI, CLBO}
F ($e_{u,7}$)= F (The influence on Medium Asset)
= {Ø}
F ($e_{u,8}$)= F (The influence on Less Important Asset)
= {CR-sS, DOCWIC}

Then the fuzzy soft set (F, E) consists of the following collection of approximations:

(F, E) = {($e_{u,1}$, ({SQLI, CLBO})), ($e_{u,2}$, ({SQLI, OSCI, CR-sS})), ($e_{u,3}$, ({SQLI, CR-sS, DOCWIC})), ($e_{u,4}$, ({OSCI, CLBO})), ($e_{u,5}$, ({SQLI})), ($e_{u,6}$, ({OSCI, CLBO})), ($e_{u,7}$, ({Ø})), ($e_{u,8}$, ({CR-sS, DOCWIC}))}

The CCEA will be applied to the first fuzzy soft part in (F, E) to decide the availability set U containing the selected values.

$$\mu_{F^dX} = \frac{1}{5}\left[\mu_x{}^{(e_{U,1})}\chi fx^{(e_{U,1})}(V_1) + \mu_x{}^{(e_{U,2})}\chi fx^{(e_{U,2})}(V_1) + \mu_x{}^{(e_{U,3})}\chi fx^{(e_{U,3})}(V_1) + \mu_x{}^{(e_{U,4})}\chi fx^{(e_{U,4})}(V_1) + \mu_x{}^{(e_{U,5})}\chi fx^{(e_{U,5})}(V_1) + \mu_x{}^{(e_{U,6})}\chi fx^{(e_{U,6})}(V_1) + \mu_x{}^{(e_{U,7})}\chi fx^{(e_{U,7})}(V_1) + \mu_x{}^{(e_{U,8})}\chi fx^{(e_{U,8})}(V_1)\right]$$

$$\mu F^dX = \frac{1}{5}[(0.5 \times 1) + (1 \times 1) + (1 \times 1) + (0.5 \times 0) + (1 \times 1) + (0.7 \times 0) + (0.5 \times 0) + (0.2 \times 0)]$$

$$\mu F^dX = \frac{1}{5}[0.5 + 1 + 1 + 1] = 0.7$$

and so on for all vulnerabilities. Then F^dX is presented as

$$F^dX = \left\{\frac{0.7}{V_1}, \frac{0.44}{V_2}, \frac{0.34}{V_3}, \frac{0.44}{V_4}, \frac{0.24}{V_5}\right\}$$

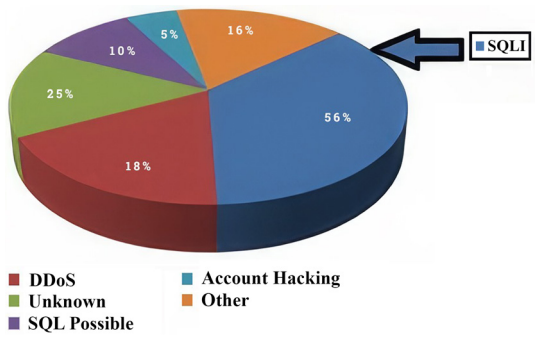
The fuzzy output decisions for these vulnerabilities (vulnerability indices) are

1) SQL Injection (SQLI) = 0.7
2) OS Command Injection (OSCI) = 0.44
3) Classic Buffer Overflow (CLBO) = 0.34
4) Cross-Site Scripting (CR-sS) = 0.44
5) Download of Code Without Integrity Check (DOCWIC)= 0.24

These five vulnerabilities are among the top 25 most dangerous software vulnerabilities in the SANS Institute and NIST lists [39, 40]. The maximum value of μF^dx = 0.7 of all the vulnerabilities means that, under this parameter (e_u), the SQL Injection vulnerability has the greatest impact on the website. As the NIST has pointed out, SQL Injection is the most dangerous software vulnerability. On the other hand, the errors that cause these vulnerabilities can directly impact the web application. To determine the importance of each parameter (u), our security experts used their collective experience and judgment to assign a weight (a fuzzy value) to each parameter. They determined these values based on the real-world impact of each factor on the overall risk of the system; these values were not taken from a specific dataset.

Figure 1 illustrates the distribution of attack techniques. The pie chart shows the different types of attack techniques and their percentage probability of occurrence. The slices represent all possible attack methods available to us, and the percentages help deduce how common each method is. More than half of all tracked attacks involve SQL vulnerability/injection, accounting for the largest share (blue, 56%). Distributed Denial-of-Service (DDoS) (red) accounts for 18% of the total. A sizeable portion of 25% falls into the "unknown" category, meaning their exact nature could not be determined. Other techniques (orange) include miscellaneous attack methods, accounting for 16% of the total. SQL Possible attacks (purple, 10%), targeting account hacking, represent 10%. The light blue slice (5%) is not labeled or clearly explained. The chart highlights the SQL injection segment (56%) with a bold black arrow and label, emphasizing its dominance in the data.

Figure 1
Distribution of attack techniques



5.2. Error index

Each error or more errors lead to one or more vulnerabilities [41, 42], which attackers use to directly control the system to read, modify, or destroy. In this section, we will calculate the error index available in the system. The error index is based on several parameters as follows:

1) The percentage of error occurrences in each software, i.e., each error may occur once or more in a specific software (the likelihood of each error being exploited).
2) The percentage effect of the error, i.e., each error may affect one or more vulnerabilities. For example, the percentage effect of error X that led to three different vulnerabilities can be higher than the percentage effect of error Y that led to one vulnerability.
3) Percentage of the dangers of error in a particular software. Every software has vulnerabilities, and one or more errors influence each vulnerability. Some of these errors affect dangerous vulnerabilities at different levels. Calculate all errors (for each error related to the system) affecting the system universe (U) described according to SANS Institute and the NIST.

$$U = (R1, R2, R3, R4, R5, R6\ R7, R8, R9, R10) \quad (9)$$

where R refers to errors. According to the SANS Institute and NIST lists, all errors are related to this type of system and are listed as follows:

1) Poor SQL commands check usernames and passwords (PSQL).
2) The program executed allows arguments to be specified within an input file or from standard input (PEAASI).
3) The code path includes a Buffer Write Operation (CPBWO).
4) Buffer is as large as parameters specify (BLS).
5) Replace unbounded copy functions with analogous ones supporting length arguments (RUCFSLA).
6) Set the session cookie to be HTTP (SSCH).
7) Assume all input is not malicious (AIM).
8) Input Validation did not consider all potentially relevant properties (IVPRP).
9) Did not use proper output encoding, escaping, and quoting (DPOEEQ).
10) Encrypt the code with a reliable encryption scheme before transmitting (ECREST).

Ten errors in the case study were related to the vulnerabilities enumerated in the previous section; otherwise, we could not quantify the security requirements.

U={PSQL, PEAASI, CPBWO, BLS, RUCFSLA, SSCH, AIM, IVPRP, DPOEEQ, ECREST}

Let E_u={e_u} be a set of decision parameters related to the above universal U "errors" according to the security experts.

E_u={$e_{u,1}$=occurrence one time; $e_{u,2}$=occurrence two times, $e_{u,3}$=occurrence three or more times, $e_{u,4}$=The influence on one vulnerability, $e_{u,5}$=The influence on two vulnerabilities, $e_{u,6}$=The influence on three or more vulnerabilities, $e_{u,7}$=The influence on very dangerous vulnerability, $e_{u,8}$=The influence on dangerous vulnerability, $e_{u,9}$=The influence on normal vulnerability}.

In the case study, nine parameters (e_u) are used; in other cases, more or less than this number of parameters and different values may be used, based on the nature of importance in the system. Then, the security experts assign these parameter values according to their importance. For instance, if error X occurs three times in the system, it should have a high value, and vice versa.

$$E = \left\{ \frac{e_{u,1}}{0.33}, \frac{e_{u,2}}{0.66}, \frac{e_{u,3}}{1}, \frac{e_{u,4}}{0.33}, \frac{e_{u,5}}{0.66}, \frac{e_{u,6}}{1}, \frac{e_{u,7}}{1}, \frac{e_{u,8}}{0.66}, \frac{e_{u,9}}{0.33} \right\} \quad (10)$$

Now, we will find the function for each parameter, F ($e_{u,1}$), which means the parameter ($e_{u,1}$) representing "occurrence one time," and according to security experts, this parameter has a value of 0.33 in this system. All errors that occur one time in this system will be part of this function, such as (ECREST, PEAASI) errors in this case study, and so on for the whole parameter.

F ($e_{u,1}$) = F(occurrence one time)
= {ECREST, PEAASI}
F ($e_{u,2}$) = F(occurrence two times)
= {CPBWO, IVPRP, DPOEEQ, AIM}
F ($e_{u,3}$) = F(occurrence three or more times)
= {PSQL, BLS, RUCFSLA, SSCH}
F ($e_{u,4}$) = F(The influence on one vulnerability)
= {PSQL, CPBWO, BLS, RUCFSLA, SSCH, AIM, DPOEEQ, ECREST}
F ($e_{u,5}$)= F(The influence on two vulnerabilities)
= {PEAASI}
F ($e_{u,6}$)= F(The influence on three or more vulnerabilities)
= {IVPRP}
F ($e_{u,7}$)= F(The influence on very dangerous vulnerability)
= {PSQL, PEAASI, IVPRP}
F ($e_{u,8}$)= F(The influence on dangerous vulnerability)
= {PEAASI, IVPRP, SSCH, DPOEEQ, AIM}
F ($e_{u,9}$) = F(The influence on normal vulnerability)
= {CPBWO, BLS, RUCFSLA, ECREST}

Then the fuzzy soft set (F, E) consists of the following collection of approximations:

(F, E) = {($e_{u,1}$, ({SQLI, CLBO})), ($e_{u,2}$, ({SQLI, OSCI, CR-sS})), ($e_{u,3}$, ({SQLI, CR-sS, DOCWIC})), ($e_{u,4}$, ({OSCI, CLBO})), ($e_{u,5}$, ({SQLI})), ($e_{u,6}$, ({OSCI, CLBO})), ($e_{u,7}$, ({Ø})), ($e_{u,8}$, ({CR- sS, DOCWIC}))}

The CCEA was applied to the first fuzzy soft part in (F, E) to determine the availability set U containing the selected values.

$$\mu_{F^dX} = \frac{1}{10}\left[\mu_x{}^{(e_{U,1})}\chi fx^{(e_{U,1})}(R_1) + \mu_x{}^{(e_{U,2})}\chi fx^{(e_{U,2})}(R_1) + \mu_x{}^{(e_{U,3})}\chi fx^{(e_{U,3})}(R_1) + \mu_x{}^{(e_{U,4})}\chi fx^{(e_{U,4})}(R_1) + \mu_x{}^{(e_{U,5})}\chi fx^{(e_{U,5})}(R_1) + \mu_x{}^{(e_{U,6})}\chi fx^{(e_{U,6})}(R_1) + \mu_x{}^{(e_{U,7})}\chi fx^{(e_{U,7})}(R_1) + \mu_x{}^{(e_{U,8})}\chi fx^{(e_{U,8})}(R_1) + \mu_x{}^{(e_{U,9})}\chi fx^{(e_{U,9})}(R_1)\right]$$

$$\mu F^dX = \frac{1}{10}[(0.33 \times 0) + (0.66 \times 0) + (1 \times 1) + (0.33 \times 1) + (0.66 \times 0) + (1 \times 0) + (1 \times 1) + (0.66 \times 0) + (0.33 \times 0)]$$

$$\mu F^dX = \frac{1}{10}[1 + 0.33 + 1] = 0.233$$

and so on for all errors. Then F^dX is represented as

$$F^dX = \left\{ \frac{0.233}{R_1}, \frac{0.265}{R_2}, \frac{0.132}{R_3}, \frac{0.166}{R_4}, \frac{0.166}{R_5}, \frac{0.199}{R_6}, \frac{0.165}{R_7}, \frac{0.332}{R_8}, \frac{0.165}{R_9}, \frac{0.099}{R_{10}} \right\}$$

The fuzzy output decisions for these errors (Error Indices) are

1) Poor SQL commands are used to check usernames and passwords (PSQL) = 0.233
2) The program be executed allows arguments to be specified within an input file or from standard input (PEAASI) = 0.265
3) Code path includes a Buffer Write Operation (CPBWO) = 0.132
4) Buffer is as large as you specify (BLS) = 0.166
5) Replace unbounded copy functions with analogous functions that support length arguments (RUCFSLA) = 0.166
6) Set the session cookie to be not only HTTP (SSCH) = 0.199
7) Assume all input is not malicious (AIM) = 0.165
8) Input Validation did not consider all potentially relevant properties (IVPRP) = 0.332
9) Did not use proper output encoding, escaping, and quoting (DPOEEQ) = 0.165
10) Encrypt the code with a reliable encryption scheme before transmitting (ECREST) = 0.099

According to the previous results, the highest value is $\mu F^d{}_X = 0.332$ among all errors; the "input validation did not consider all potentially relevant properties" error is the most influential on the website under this parameter (E_u).

In the next section, the security and requirement indexes will be evaluated based on the previously calculated error indices and vulnerability indices.

5.3. Security index

After calculating the vulnerability and error indices associated with the web application, the next step in this case study is to calculate the SI. Finally, we connect the SI contents with the security requirements to obtain our result, the security requirements index. There is a relationship between the error index and vulnerability index; this relationship gives us the SI. Using Equation (5), we can calculate the SI. We need the vulnerabilities and error indices calculated in the last two sub-sections to calculate the SI.

Vulnerabilities Index:

1) SQL Injection (SQLI) = 0.7
2) OS Command Injection (OSCI) = 0.44
3) Classic Buffer Overflow (CLBO) = 0.34
4) Cross-Site Scripting (CR-sS) = 0.44
5) Download of Code Without Integrity Check (DOCWIC)= 0.24

Error index:

1) Poor SQL commands are used to check usernames and passwords (PSQL) = 0.233
2) The program be executed allows arguments to be specified within an input file or from standard input (PEAASI) = 0.265
3) Code path includes a Buffer Write Operation (CPBWO) = 0.132
4) Buffer is as large as you specify (BLS) = 0.166
5) Replace unbounded copy functions with analogous functions that support length arguments (RUCFSLA) = 0.166
6) Set the session cookie to be not only HTTP (SSCH) = 0.199
7) Assume all input is not malicious (AIM) = 0.165
8) Input Validation did not consider all potentially relevant properties (IVPRP) = 0.332
9) Did not use proper output encoding, escaping, and quoting (DPOEEQ) = 0.165
10) Encrypt the code with a reliable encryption scheme before transmitting (ECREST) = 0.099

Each vulnerability consists of one or more errors. To calculate the SI for any security requirement, we must first find the relationships between vulnerabilities and errors and then use Equation (5) using the vulnerability index and error index to find the SI as shown below:

1) SQLI ={PSQL, PEAASI, IVPRP}
2) OSCI ={PEAASI, IVPRP}
3) CLBO ={CPBWO, BLS, RUCFSLA}
4) CR-sS ={SSCH, AIM, IVPRP, DPOEEQ}
5) DOCWIC ={ECREST}

That means that the "SQL Injection (SQLI)" vulnerability consists of "Poor SQL commands are used to check user names and passwords (PSQL)" error, "The program be executed allows arguments to be specified within an input file or from standard input (PEAASI)" error, "Input validation did not consider all potentially relevant properties (IVPRP)" error, and so on for all vulnerabilities mentioned above. Now we calculate the SI for each vulnerability using Equation (5) as follows:

$$SI^{Vi} = \sum_{i,j \in I} V_i E_j / j$$

$$SI^{SQLI} = \frac{SQLI \times PSQL + SQLI \times PEAASI + SQLI \times IVPRP}{3}$$

$$= \frac{(0.7 \times 0.233) + (0.7 \times 0.265) + (0.7 \times 0.332)}{3}$$

$$= \frac{0.1631 + 0.1855 + 0.2324}{3}$$

$$SI^{SQLI} = \frac{0.581}{3}$$

$$SI^{SQLI} = 0.1936$$

and so on for all SI.

Now we have an SI for all the vulnerabilities; fuzzy numbers mean that when the number approaches 1, the seriousness of this vulnerability increases; thus, it should be removed as a high priority, and vice versa. We will arrange the vulnerabilities according to their severity as follows:

Security Index:

1) SQL Injection (SQLI) = 0.1936
2) OS Command Injection (OSCI) = 0.1313
3) Cross-Site Scripting (CR-sS) = 0.0947
4) Classic Buffer Overflow (CLBO) = 0.0525
5) Download of Code Without Integrity Check (DOCWIC) = 0.0237

5.4. Security requirements index

All SI contents will be connected to all security requirements covering all software/business requirements (Assets). Therefore, we should connect each SI with a related security requirement; sometimes, such a security requirement connects with two or more SI contents.

1) Command Injection Flaws: (SQLI) and (OSCI) = 0.3249
2) Data and Input Validation: (SQLI) and (OSCI) = 0.3249
3) Command Injection Flaws: (OSCI) = 0.1313
4) Cross-Site Scripting (XSS): (CR-sS) = 0.0947
5) Buffer Overflows: (CLBO) = 0.0525
6) Authentication: (DOCWIC) = 0.0237

5.5. Comparison with other methods

This paper presented a method for evaluating how well security requirements are met in software development using fuzzy soft set theory, a mathematical tool designed to manage uncertainty and ambiguity. This approach stands out because it focuses on addressing security requirements from the start of the SDLC, unlike many traditional methods that only address security issues at the later stages. An example of this is Somerville's modified framework, which targets security through the RESMM, which targets security in the requirements engineering. Similarly, the DIBR II employs the fuzzy logic approach for MCDM to better assess security needs [42]. Both methodologies offer multiple ways to enhance software security. However, they differ in how and when they are used. The fuzzy soft set theory method is noteworthy due to its role in improving decision-making in the early stages of the SDLC. This enhancement may help prevent vulnerabilities from escalating into a problem.

The maturity model for secure requirements engineering proposed in Niazi et al. [29] allows organizations to assess how deeply security has been integrated into their processes. Assessing an organization's performance in managing its security requirements involves mapping the organization's level of maturity, identifying gaps and making recommendations for improvement. While this method can be helpful for strategic or management considerations, case-by-case inquiries do not provide the necessary computational or quantitative criteria to assess or order any particular security requirement. Essentially, their model examines the quality of the entire process. However, it does not provide a framework for decision-making in situations where security information is uncertain or incomplete. On the contrary, this paper proposes a fuzzy soft set–based approach that allows for quantitative measurements of security requirements to assist practitioners in making informed and objective choices rather than relying solely on qualitative measurements of maturity. The proposed method is more realistic for software engineers, who often encounter ambiguous or contradictory security information in practice.

Similarly, Tešić et al. [27] introduced an improved MCDM technique using fuzzy logic and the DIBR II method (with triangular fuzzy numbers). Their work aims to improve the accuracy of decisions made under uncertain conditions in areas such as management and industrial prioritization. While it effectively captures uncertainty and relationships between factors, it does not involve software security or requirements engineering. However, this document goes beyond these previous works by applying fuzzy soft set theory to quantify security requirements themselves, since these are inherently uncertain and naturally depend on the context. This model improves the clarity and flexibility of evaluating security risks and can be easily applied to actual

software development processes.

This paper builds on previous contributions and expands on existing research findings, including the work of Niazi et al. [29], which explored organizational maturity in the context of 50 security practices, and the work of Tešić et al. [27], which developed a general decision-making approach under uncertainty. The outcome is a specialized decision-support model that can address the security requirements of modern software systems more comprehensively and effectively.

6. Conclusion

The fuzzy soft set and CCEA provide many advantages in software security. Security requirements engineering allows for determining the security requirements with greater precision and value. This method aids security professionals and development teams in comprehending, evaluating, and prioritizing security requirements throughout a project. Furthermore, it helps to achieve unique and accurate security requirements. As a result, the developed software will be more secure against threats.

It is another important aspect that the technique increases motivation and involvement of contributors as they can see clearly how and where their contribution impacts software security. It improves security awareness but also develops a more security-prone culture in the software industry.

It is a well-known fact that a number of factors contribute to the success of a software development process. One of the most essential of these factors involves ensuring the security requirements are addressed thoroughly in order to avoid vulnerabilities. In the future work, we plan to extend research on the early requirements phase elicitation of security requirements. We also plan to produce a broader range of test cases for high-impact vulnerabilities to ensure that they are incorporated into the design. This will result in improved product quality and testing efficiency.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data are available from the corresponding author upon reasonable request.

Author Contribution Statement

Dyala Ibrahim: Conceptualization, Software, Resources, Data curation, Writing – review & editing, Supervision, Project administration. **Omar Isam Al Mrayat:** Conceptualization, Methodology, Software, Resources, Data curation, Writing – review & editing, Supervision, Project administration. **Lial Raja Alzabin:** Software, Resources, Data curation, Visualization. **Ahmad Alajarmeh:** Validation, Visualization. **Ahmad Reda Alzighaibi:** Validation, Formal analysis, Writing – original draft. Hasan Hashim: Investigation. **El-Sayed Atlam:** Investigation. **Malik Jawarneh:** Investigation.

References

[1] Madan, B. B., Goševa-Popstojanova, K., Vaidyanathan, K., & Trivedi, K. S. (2004). A method for modeling and quantifying the security attributes of intrusion tolerant systems. *Performance Evaluation*, *56*(1–4), 167–186. https://doi.org/10.1016/j.peva.2003.07.008

[2] Shukla, A., Katt, B., & Yamin, M. M. (2023). A quantitative framework for security assurance evaluation and selection of cloud services: A case study. *International Journal of Information Security*, *22*(6), 1621–1650. https://doi.org/10.1007/s10207-023-00709-8

[3] Fabian, B., Gürses, S., Heisel, M., Santen, T., & Schmidt, H. (2010). A comparison of security requirements engineering methods. *Requirements Engineering*, *15*(1), 7–40. https://doi.org/10.1007/s00766-009-0092-x

[4] Meridji, K., Al-Sarayreh, K. T., Abran, A., & Trudel, S. (2019). System security requirements: A framework for early identification, specification and measurement of related software requirements. *Computer Standards & Interfaces*, *66*, 103346. https://doi.org/10.1016/j.csi.2019.04.005

[5] Venson, E., Clark, B., & Boehm, B. (2024). The effects of required security on software development effort. *Journal of Systems and Software*, *207*, 111874. https://doi.org/10.1016/j.jss.2023.111874

[6] de Win, B., Scandariato, R., Buyens, K., Grégoire, J., & Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, *51*(7), 1152–1171. https://doi.org/10.1016/j.infsof.2008.01.010

[7] Hamdani, S. W. (2023). *Framework for assessing information system security posture risks*. Master's Thesis, The University of Western Ontario.

[8] Fujs, D., Vrhovec, S., & Vavpotič, D. (2023). Balancing software and training requirements for information security. *Computers & Security*, *134*, 103467. https://doi.org/10.1016/j.cose.2023.103467

[9] Mubarkoot, M., Altmann, J., Rasti-Barzoki, M., Egger, B., & Lee, H. (2023). Software compliance requirements, factors, and policies: A systematic literature review. *Computers & Security*, *124*, 102985. https://doi.org/10.1016/j.cose.2022.102985

[10] Hnaini, H., Mazo, R., Vallejo, P., Lopez, A., Champeau, J., & Galindo, J. (2024). SECRET: A new security requirements specification template. In *Information Technology and Systems*, *2*, 235–246. https://doi.org/10.1007/978-3-031-54256-5_22

[11] Alotaibi, B. (2023). A survey on industrial internet of things security: Requirements, attacks, AI-based solutions, and edge computing opportunities. *Sensors*, *23*(17), 7470. https://doi.org/10.3390/s23177470

[12] Bulusu, S. T., Laborde, R., Wazan, A. S., Barrère, F., & Benzekri, A. (2018). A requirements engineering-based approach for evaluating security requirements engineering methodologies. In *Information Technology–New Generations: 15th International Conference on Information Technology*, 517–525. https://doi.org/10.1007/978-3-319-77028-4_67

[13] Janisar, A. A., Shafee bin Kalid, K., Sarlan, A. B., & Maiwada, U. D. (2024). Software development teams knowledge and awareness of security requirement engineering and security requirement elicitation and analysis. *Procedia Computer Science*, *234*, 1348–1355. https://doi.org/10.1016/j.procs.2024.03.133

[14] El Ghazi El Houssaïni, S., Maskani, I., & Boutahar, J. (2021). A security requirement engineering case study: Challenges and lessons learned. In *Intelligent Computing: Proceedings of the 2021 Computing Conference*, *3*, 761–783. https://doi.org/10.1007/978-3-030-80129-8_52

[15] Khanneh, S., & Anu, V. (2022). Security requirements prioritization techniques: A survey and classification framework. *Software*, *1*(4), 450–472. https://doi.org/10.3390/software1040019

[16] Roy, A. R., & Maji, P. K. (2007). A fuzzy soft set theoretic approach to decision making problems. *Journal of Computational and Applied Mathematics*, *203*(2), 412–418. https://doi.org/10.1016/j.cam.2006.04.008

[17] Jaleel, A. (2022). WASPAS technique utilized for agricultural robotics system based on Dombi aggregation operators under bipolar complex fuzzy soft information. *Journal of Innovative Research in Mathematical and Computational Sciences*, *1*(2), 67–95.

[18] Özer, Ö. (2022). Hamacher prioritized aggregation operators based on complex picture fuzzy sets and their applications in decision-making problems. *Journal of Innovative Research in Mathematical and Computational Sciences*, *1*(1), 33–54.

[19] Feng, F., Jun, Y. B., Liu, X., & Li, L. (2010). An adjustable approach to fuzzy soft set based decision making. *Journal of Computational and Applied Mathematics*, *234*(1), 10–20. https://doi.org/10.1016/j.cam.2009.11.055

[20] Alcantud, J. C. R. (2016). A novel algorithm for fuzzy soft set based decision making from multiobserver input parameter data set. *Information Fusion*, *29*, 142–148. https://doi.org/10.1016/j.inffus.2015.08.007

[21] Hussain, A., Ullah, K., Pamucar, D., Haleemzai, I., & Tatić, D. (2023). Assessment of solar panel using multiattribute decision-making approach based on intuitionistic fuzzy Aczel Alsina Heronian mean operator. *International Journal of Intelligent Systems*, *2023*(1), 6268613. https://doi.org/10.1155/2023/6268613

[22] Mahmood, T. (2020). A novel approach towards bipolar soft sets and their applications. *Journal of Mathematics*, *2020*(1), 4690808. https://doi.org/10.1155/2020/4690808

[23] Selvaraj, M., & Vadivel, R. (2024). An approach based on hexagram model for quantifying security risks with Performance Key Indicators (PKI). *Computers and Electrical Engineering*, *116*, 109135. https://doi.org/10.1016/j.compeleceng.2024.109135

[24] Nhlabatsi, A., Khan, K. M., Hong, J. B., Kim, D. S., Fernandez, R., & Fetais, N. (2023). Quantifying satisfaction of security requirements of cloud software systems. *IEEE Transactions on Cloud Computing*, *11*(1), 426–444. https://doi.org/10.1109/TCC.2021.3097770

[25] Andrade, R., Torres, J., Ortiz-Garcés, I., Miño, J., & Almeida, L. (2023). An exploratory study gathering security requirements for the software development process. *Electronics*, *12*(17), 3594. https://doi.org/10.3390/electronics12173594

[26] Al-Matouq, H., Mahmood, S., Alshayeb, M., & Niazi, M. (2020). A maturity model for secure software design: A multivocal study. *IEEE Access*, *8*, 215758–215776. https://doi.org/10.1109/ACCESS.2020.3040220

[27] Tešić, D., Božanić, D., & Khalilzadeh, M. (2024). Enhancing multi-criteria decision-making with fuzzy logic: An advanced defining interrelationships between ranked II method incorporating triangular fuzzy numbers. *Journal of Intelligent Management Decision*, *3*(1), 56–67. https://doi.org/10.56578/jimd030105

[28] Alojaiman, B. (2023). A multi-criteria decision-making process for the selection of an efficient and reliable IoT application. *Processes*, *11*(5), 1313. https://doi.org/10.3390/pr11051313

[29] Niazi, M., Saeed, A. M., Alshayeb, M., Mahmood, S., & Zafar, S. (2020). A maturity model for secure requirements engineering. *Computers & Security*, *95*, 101852. https://doi.org/10.1016/j.cose.2020.101852

[30] Muhammad, A., Siddique, A., Mubasher, M., Aldweesh, A., & Naveed, Q. N. (2023). Prioritizing non-functional requirements in agile process using multi criteria decision making analysis. *IEEE Access*, *11*, 24631–24654. https://doi.org/10.1109/ACCESS.2023.3253771

[31] Agrawal, A., Alenezi, M., Khan, S. A., Kumar, R., & Khan, R. A. (2022). Multi-level Fuzzy system for usable-security assessment. *Journal of King Saud University - Computer and Information Sciences*, *34*(3), 657–665. https://doi.org/10.1016/j.jksuci.2019.04.007

[32] Paik, B., & Mondal, S. K. (2022). Introduction to soft-cryptosystem and its application. *Wireless Personal Communications*, *125*(2), 1801–1826. https://doi.org/10.1007/s11277-022-09635-9

[33] Ali, Z. (2022). Decision-making techniques based on complex intuitionistic fuzzy power interaction aggregation operators and their applications. *Journal of Innovative Research in Mathematical and Computational Sciences*, *1*(1), 107–125.

[34] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

[35] Molodtsov, D. (1999). Soft set theory—First results. *Computers & Mathematics with Applications*, *37*(4–5), 19–31. https://doi.org/10.1016/S0898-1221(99)00056-5

[36] Cagman, N., & Enginoglu, S. (2011). FP-soft set theory and its applications. *Annals of Fuzzy Mathematics and Informatics*, *2*(2), 219–226.

[37] Yang, J., & Yao, Y. (2020). Semantics of soft sets and three-way decision with soft sets. *Knowledge-Based Systems*, *194*, 105538. https://doi.org/10.1016/j.knosys.2020.105538

[38] Alcantud, J. C. R. (2022). The semantics of N-soft sets, their applications, and a coda about three-way decision. *Information Sciences*, *606*, 837–852. https://doi.org/10.1016/j.ins.2022.05.084

[39] Guttman, B., & Roback, E. (1995). *An introduction to computer security: The NIST handbook* (NIST Special Publication 800-12). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-12

[40] Pescatore, J. (2021). *SANS 2021 top new attacks and threat report* [White paper]. SANS Institute. https://information.rapid7.com/rs/411-NAK-970/images/Whitepaper_Attack%26Threat-Report.pdf

[41] Alcantud, J. C. R., Khameneh, A. Z., Santos-García, G., & Akram, M. (2024). A systematic literature review of soft set theory. *Neural Computing and Applications*, *36*(16), 8951–8975. https://doi.org/10.1007/s00521-024-09552-x

[42] Wen, S.-F., Shukla, A., & Katt, B. (2022). Developing security assurance metrics to support quantitative security assurance evaluation. *Journal of Cybersecurity and Privacy*, *2*(3), 587–605. https://doi.org/10.3390/jcp2030030

How to Cite: Ibrahim, D., Al Mrayat, O. I., Alzabin, L. R., Alajarmeh, A., Alzighaibi, A. R., Hashim, H., Atlam, E. S., & Jawarneh, M. (2025). Measuring Satisfaction of Security Requirements Using Fuzzy Soft Set Theory. *Journal of Computational and Cognitive Engineering*. https://doi.org/10.47852/bonviewJCCE52026871