ANALYTIC HIERARCHY PROCESS-BASED FUZZY MEASUREMENT TO QUANTIFY VULNERABILITIES OF WEB APPLICATIONS

Mohammad Shojaeshafiei[1], Letha Etzkorn[1] and Michael Anderson[2]

[1]Department of Computer Science, The University of Alabama in Huntsville, Huntsville, USA
[2] Department of Civil and Environmental Engineering, The University of Alabama in Huntsville, Huntsville, USA

ABSTRACT

Much research has been conducted to detect vulnerabilities of Web Applications; however, these never proposed a methodology to measure the vulnerabilities either qualitatively or quantitatively. In this paper, a methodology is proposed to investigate the quantification of vulnerabilities in Web Applications. We applied the Goal Question Metrics (GQM) methodology to determine all possible security factors and sub-factors of Web Applications in the Department of Transportation (DOT) as our proof of concept. Then we introduced a Multi-layered Fuzzy Logic (MFL) approach based on the security sub-factors' prioritization in the Analytic Hierarchy Process (AHP). Using AHP, we weighted each security sub-factor before the quantification process in the Fuzzy Logic to handle imprecise crisp number calculation.

KEYWORDS

vulnerability, web applications, analytic hierarchy process, fuzzy logic, goal question metrics, cybersecurity

1. INTRODUCTION

A large number of users for daily work and communications are relying on Web Applications. These users can potentially be targeted by cyberattacks either due to having interaction with personal computers or operating in a large network of a susceptible corporation. In most circumstances, if the security structure of the system is not strict enough, expensive consequences follow. The total average cost of a data breach on organizations in 2019 was $ 3.92 million based on the research conducted by IBM [1]. Cross-site Scripting (XSS) and SQL injection are the most destructive and pervasive types of attacks that threaten organizations' secret information through web applications' vulnerabilities. According to statistics [2] Web Applications have become the number one target for vulnerability exploitation such that 46% of web applications in 2019 suffered from critical vulnerability, 78% had medium security vulnerability, 30% were vulnerable to XSS and 4 out of 5 Web Applications contain configuration errors. The lack of a comprehensive approach to quantify and understand the vulnerabilities of Web Applications has always been a significant issue. Much research has been conducted in this area and several valuable methodologies have been proposed either to bolster the security of Web Applications or to detect potential vulnerabilities [3][4]. Unfortunately, from the perspective of Web Applications, vulnerability quantification is very intricate and requires strenuous work to obtain a numerical value due to the qualitative nature of vulnerabilities in the system (i.e., "very good", "medium" or "very bad"). To the best of our knowledge, this is the first work that attempts to investigate Web Applications' vulnerability quantification through the Fuzzy Logic approach [5]. We accomplish this through assigning an uneven weight for each fuzzy rule in the Fuzzy Inference System (FIS) processes. The research undertaken for this study has been driven by three major purposes: (1) define a standard framework for Web Application security factors, (2) quantify the vulnerabilities of Web Applications by determining the metrics for each sub-factor in a backward manner using a Fuzzy Logic approach, and (3) apply the Analytic Hierarchy Process (AHP) to prioritize each fuzzy rule before quantification. In order to meaningfully articulate the vulnerability measurement processes, we have studied the Department of Transportation (DOT) as a proof of concept using an in-depth analysis to comprehend the different aspects of Web Applications' vulnerabilities that are required for a meticulous security consideration. To do this, we applied the Goal Question Metrics (GQM) methodology [6] to define and design all factors and sub-factors of Web Applications for vulnerability quantification. The outputs of GQM are a security factor and sub-factor graph tree that needs to be evaluated using the Fuzzy Logic approach to calculate the appropriate metric for each successor (sub-factor) derived from a predecessor (main factor) of the tree. The important point here is to determine a meaningful weight for each rule using the AHP [7] for prioritizing the security sub-factors in the steps prior to the Defuzzification process to obtain a crisp value for vulnerability. In other words, AHP helps us to make a rational judgment about the prioritization of security sub-factors in Web Applications. We use a pair-wise comparison [7] of acquired knowledge from the graph tree to determine the importance of each criterion in the security framework.

The rest of the paper is structured as follows: Section 2 addresses the related work in Web Applications' vulnerability from different aspects of security. Section 3 describes the methodology using GQM, security factors and sub-factors, AHP, and MFL. Section 4 addresses the implementation using MFL with the comparison of vulnerability quantification with/without uneven weights.

2. BACKGROUND

2.1. Vulnerabilities in Web Applications

The degree to which the adversary is capable of taking control of the system or some level of the hosting server due to misconfiguration or the available bugs in the software is a vulnerability in Web Applications. There has been some research from other researchers to analyze [8], scan [9], automate [10], predict [11], and detect [12] vulnerabilities of Web Applications either statically or dynamically. While the state of the art emphasizes different types of predominant attacks (e.g. XSS) to detect the vulnerabilities of Web Applications [13][14], the lack of research on vulnerability measurement (quantitatively and qualitatively) is egregious at higher levels.

2.2. Literature Review

Felmetsger et al. [15] researched automated detection of vulnerabilities in Web Applications using the dynamic analysis to infer a pattern for behavioral specifications. They filtered the learned specifications from leveraging knowledge about the typical execution paradigm of Web Applications to pare false positives as much as possible. They developed a tool called Walter to discern a program path to check if all input symbols violate the behavioral specifications which derives program invariants for Web Applications to identify unknown application logic flaws.

Jovanovic et al. [16] presented an automated solution for vulnerability detection of Web Applications to overcome the obstacle of error-prone manual code review. In order to express the vulnerabilities of Web Applications through static source code, they applied an interprocedural and context-sensitive data flow to detect flaws in a program. They proposed a tool in an open-source implementation, Pixy, to detect SQL, XSS, and command injection types of vulnerabilities. For experimental validation, they discovered 15 previously unknown vulnerabilities and reconstructed 36 known vulnerabilities (one false positive on average).

Huang et al. [17] conducted research on Web Applications security assessment that led to vulnerabilities due to poor coding practices. They designed a crawler interface to mimic the functionalities of the web browser to test Web Applications. To find all data entry points to begin with they used a reverse engineering methodology phase. The entry points were then applied as fault injection processes in the Non-Rete (NRE) algorithm to remove false-negatives. They used software testing techniques in a tool called WAVES to analyze the resulting pages. The authors claim it has the capability to handle cookie poisoning, parameter tampering, buffer overflow, and hijacking.

Sonmez [18] addressed a study to define security qualitative metrics for Web Applications. She provided a list of web application development criteria with 230 qualitative metrics in order to facilitate the Open Web Application Security Project (OWASP) compliance analysis. The author claims if the OWASP compliance is more discernible, it may be helpful to qualitatively measure the attack surface in Web Applications.

Priya et al. [19] worked on the Rational Unified treatment Process (RUP). They implemented and developed Rational Unified WVA to assess Web Applications' vulnerabilities in a software development process framework. This is an iterative approach to develop a framework for any object-oriented or web-oriented program that brings about vulnerability detection, report, and vulnerability remediation accordingly due to lack of security in the program that allows the intruder or adversary to gain unauthorized access to the system. The WVA has four main steps: discovery, audit, exploit, and evasion. These steps were performed in sequence to provide appropriate vulnerability reports in the testbed.

Philipsen [20] in 2005 provided a report on Web Applications' vulnerability assessment in order to discover and propose mitigations for security issues. The author indicated that three main reasons make Web Applications vulnerable to attacks as follows: (1) design with lack of security consideration at early stages (2) reliance on other architecture components such as depending on the connection of Web Applications to a back-end database that provides all rights on the database, and (3) abundance of programming languages that makes it complex to provide an infrastructure with a standard security framework. Furthermore, the author presented several vulnerability mitigation recommendations in order to ease the organizations' security standards and policies to comply with if needed.

Luh et al. [30] proposed a model, PenQuest, to illustrate a complete view of information system attacks and their mitigation. This model provides a time-enabled attacker and defender behavior as part of NIST sp800-53. It maps attack patterns to counterpart models through data-centric procedures to present a gamified approach, a game based on a multi-tiered attacker and a defender model that is useful for personnel training, risk assessment in IT infrastructure. The model is composed of Service, Information, Event, and PenQuest Core Layers in the System. The relationship between these layers creates a concrete class definition to outline the game theories. It introduces a hybrid model that can be categorized based on some principles of game theory as follows: non-cooperative nonzero-sum game, asymmetric strategy, dynamic elements, imperfect information, Bayesian formula, and finite. The authors claim that the model is very flexible and enables domain experts to enlarge the system by adding new actors and features. This model can be an appropriate means to develop the foundation for target attack and especially an anomaly detection interpretation for intrusion detection systems.

Shafee et al. [31] proposed an intrusion detection model that provides a mimic learning strategy of learning. It adopts the process of using intrusion detection knowledge to be trained in a teacher model on private data and then transferred to a student model to be able to share knowledge without sharing the original data. The model framework is composed of Intrusion Detection Agencies, Organizations, and Intrusion Detection Gateways to provide three modules to reach the goal as follows: (1) Monitoring Modules, (2) Feature Extraction Modules and, (3) Classifier Module. These modules are completely independent of the teacher and the student modules. The teacher module chooses the best classifier with the best performance through the evaluation of each classifier. This classifier will pass through the processes of training on the original data set. When the teacher model finishes labeling the unlabeled data, the same process is used to produce the student model. The critical step is that after constructing the teacher models private data should be passed to the student model. At the end of the experiment, the authors' finding on the identical performance of both the teacher and the student model showed the successful measurement of the processes in mimicking learning techniques to transfer knowledge using unlabeled public data.

3. Methodology and the Proposed Approach

3.1. Goal Question Metrics

Since the main goal of our study is vulnerability quantification in Web Applications, first we are required to construct appropriate metrics to achieve the goal. A well-known methodology to define, design, and map such metrics to the goal is the Goal Question Metrics (GQM) methodology [6]. GQM handles the process of developing and maintaining metrics in a framework (here, a security framework) based on three substantial steps: (1) conceptual level: defines the desired goal(s) including the purposes of each goal for the project (2) operational level: designs meaningful and comprehensive question(s) as an operational mechanism that relates the goal(s) to the appropriate metrics and (3) quantitative level: calculates metrics based on answers to the related questions in each level in order to map the metrics to the goal(s).

As shown in Figure 1 a goal may contribute to generating one or more questions and these questions contribute to the metrics as data for the generated questions [21].

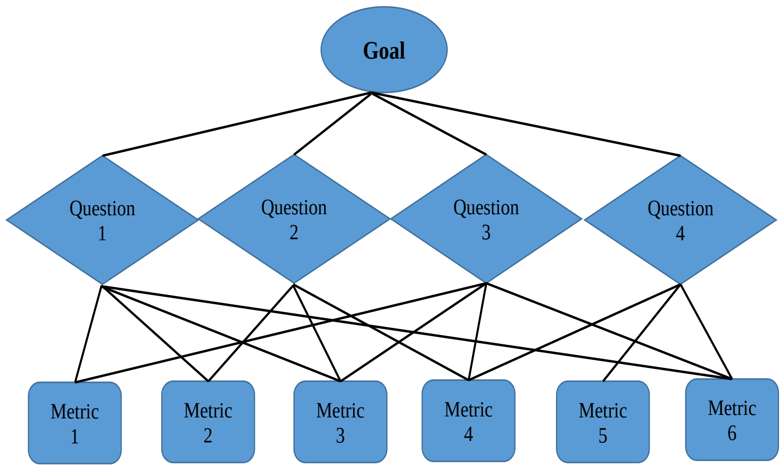


Figure 1. Basic structure of the Goal Question Metrics

One of the primary advantages of applying GQM to our research is that it is a de facto standard used by software engineers to determine software quality metrics [22] [23] [24]. Moreover, GQM is a multipurpose methodology that not only decreases project cost, risk and cycle time, it also increases project quality (to achieve the goals) because it accomplishes only those attributes that are directly related to the project objectives. Especially, in our case, we can rely on it as a suitable procedure to map elicitated cybersecurity requirements to the security goals [25].

In this paper, we consider the security requirements of Web Applications in DOT to provide a fundamental platform in order to construct our GQM framework based on essential security factors and sub-factors. The analysis of security factors and Sub-factors leads us to generate a graph tree that comprises the root (Web Applications) and the nodes (sub-factors) that pave the ground for further security analysis such as detecting and eliminating those unnecessary sub-factors that overlap conceptually. In the next section 3.2, before moving forward to the quantification processes, we will express how to articulate security factors and sub-factors to design security questions from the GQM procedure.

3.2. Security Factors and Sub-factors

The processes of security factors' extraction are onerous tasks to be captured in an organized theme. It can be observed that security factors' determination in Web Applications is not intricate, but what makes it cumbersome is (1) organizing the factors that cover multiple aspects of the security in Web Applications as part of crucial components in security of an organization, (2) designing components that directly reflect the quality of Web Applications' security in the system. These security factors and sub-factors should be able to represent the essence of security quality components such as functionality, reliability, usability, efficiency, portability, and maintainability. Therefore, each factor is considered individually to assure that it contributes sufficient broadly to include security components optimally for the design. For that, a comprehensive list of security factors and sub-factors based on their role in Web Applications is researched. As shown in Figure 2 we generated a tree graph that represents all security factors and sub-factors to quantify the vulnerabilities of Web Applications in DOT.

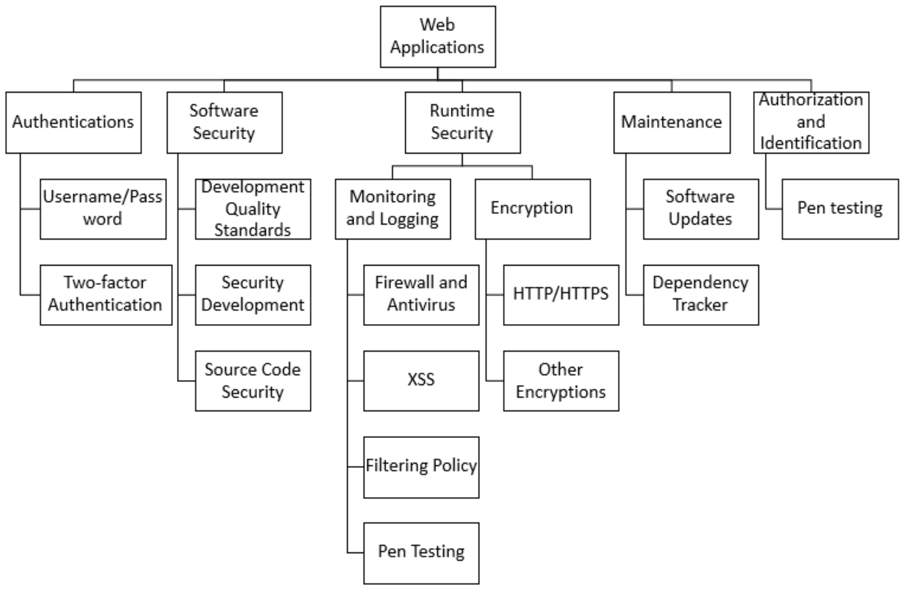


Figure 2. Security factors and sub-factors of Web Applications in DOT

We considered all aspects of security issues in DOT and after a thorough analysis, we extracted the most highlighted components that are influential either directly or indirectly to the vulnerability of DOT-related Web Applications. We attempted to eliminate the sub-factors that overlap other sub-factors; however, in some cases accepting redundancy at some points is inevitable. For instance, considering the sub-factor "Penetration Testing" in both "Runtime Security " and "Authorization and Identification " is significantly important and there is no way to remove it from one side, rather it must be accepted as necessary on both sides.

Since the final goal for the proof of concept of this research is to obtain a numerical value for vulnerabilities of Web Application in DOT, we divided the whole problem into several sub-problems based on the main security factors at each level. This technique makes the measurement processes more plausible for the application of Fuzzy Logic methodology. The sub-problems should be solved in a backward manner from the very last child node towards the parent node. When all sub-problems are solved, we apply aggregation to acquire the value of each main problem. Then each main problem comes to another aggregation equation to contribute the final result of the problem.

We chose one of the main factors (main problem) randomly as an example to address how it demonstrates the entire metric aggregation process, provided that metrics have already been calculated. The metric measurement processes should be accomplished with the Fuzzy Logic approach based on the priorities that AHP provides to the if-then rules in FIS. Figure 3 depicts the characteristics of backward-manner aggregation to achieve "Runtime Security" as one of the main factors of Web Applications in DOT.

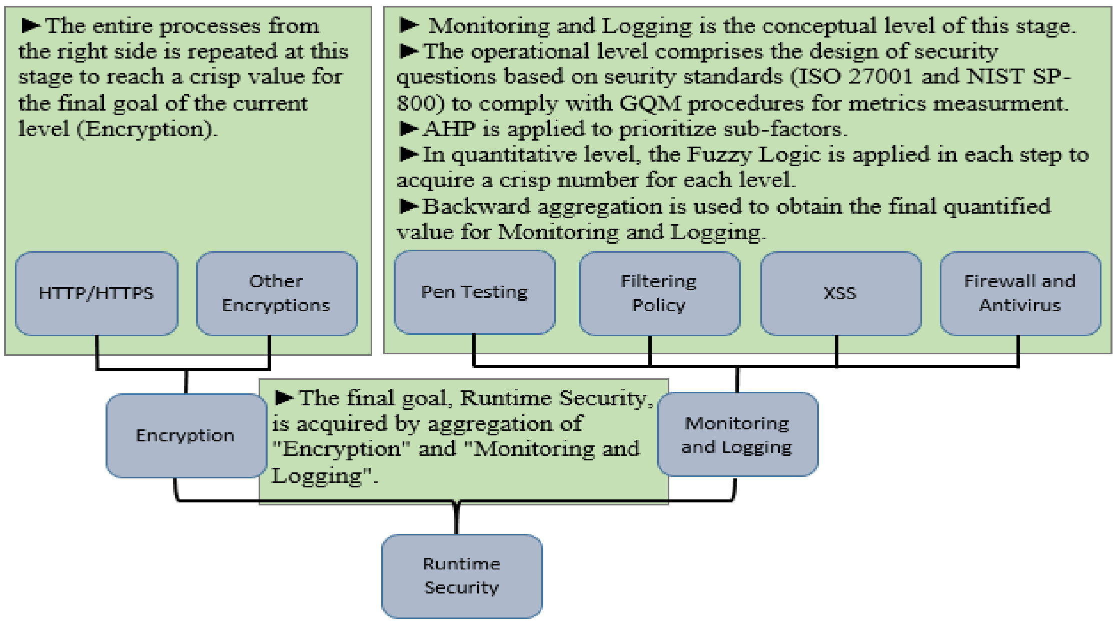

►The entire processes from the right side is repeated at this stage to reach a crisp value for the final goal of the current level (Encryption).
► Monitoring and Logging is the conceptual level of this stage.
►The operational level comprises the design of security questions based on seurity standards (ISO 27001 and NIST SP-800) to comply with GQM procedures for metrics measurment.
►AHP is applied to prioritize sub-factors.
►In quantitative level, the Fuzzy Logic is applied in each step to acquire a crisp number for each level.
►Backward aggregation is used to obtain the final quantified value for Monitoring and Logging.
HTTP/HTTPS
Other Encryptions
Pen Testing
Filtering Policy
XSS
Firewall and Antivirus
Encryption
►The final goal, Runtime Security, is acquired by aggregation of "Encryption" and "Monitoring and Logging".
Monitoring and Logging
Runtime Security

Figure 3. Factors and sub-factors of Runtime Security based on GQM's steps

As aforementioned in the operational level of GQM, we are required to design question(s) to calculate the metrics at each level. Hence, we scrutinized all security aspects of Web Applications in DOT to provide a meticulously designed questionnaire that overlays all potential vulnerability concerns. To affirm that we avoid any risk of being involved with irrelevant questions, we mapped each question to security standards such as NIST SP-800 53 and ISO-27001 at each level of security factors. The questionnaire is specifically designed to be answered in a real-world environment by DOT network security experts. For each security sub-factor, we assigned one or more questions. The number of questions for each factor depends on the importance and role of that factor in DOT. For example, if the current operation in DOT puts more emphasis on "Runtime Security" than "Maintenance" then there will evidently be more questions related to the former than the latter.

The questionnaire was sent to 2000 experts at the Department of Transportation. Twenty-six experts have replied. Some typical questions from the questionnaire are as follows:
For the Availability quality factor:

• Does DOT make sure security mechanisms and redundancies implemented to protect equipment from utility service outages (e.g., power failures, network disruptions, etc.)
• What percentage of DOT's equipment has security mechanisms and redundancy to protect equipment from utility service outages built-in?
• Does DOT implement network redundancy in its system?
• Does DOT perform industry-standard monitoring and alerting on devices that process and store sensitive information?
• How frequently does DOT back data up?
For the Integrity quality factor:
• How often does your agency inspect and account for data quality errors and associated risks?
• Which of the following options best describe the Operating System(OS) updates management and installation in your agency's computers?
• How are third-party software updates (Adobe, Java, etc.) installed on your agency's computers?
• How often does your agency verify that software suppliers adhere to industry standards for software development life cycle security?

After further consideration, we categorized questions based on their appropriate security factors into the different blocks of questions to be able to easily recognize security attributes of the parent node on the system for vulnerability measurement processes. Table 1 shows an example of security questions designed for Web Applications in DOT.

Table 1. Security question examples for each factor

Security Factor	Question
Authorization and Identification	How often does DOT control the authorization restriction policies for users/employees who have interaction with the company's web applications?
Authentication	How often does DOT require/ remind regular web applications' users to have two-factor authentication in order to login to the system?
Software Security	How often does DOT have controls in place to ensure that standards of quality are being met for all software development?
Runtime Security	How software-based firewall (windows or third party etc.) is running on DOTs computers to protect against the internal spread of a worm virus or hostile attack from another computer on the network?
Maintenance	How often does DOT use any dependency tracker/map for its web application?

As a result, we constructed a standard security framework of Web Applications, such that each security sub-factor is directly derived from either NIST SP800-53 or ISO 27001-2013 guidelines with the least number of overlap in security measures.

3.3. Security Sub-factor Prioritization using AHP

To achieve more precise vulnerability quantification, the prioritization of security sub-factor assessment is the primary objective of this section. For that, AHP is the most appropriate tool to assist us to evaluate variables (security sub-factors in this case) that are very difficult to be assigned a value. Whereas in each level of vulnerability assessment we encounter multiple attributes of a factor, decision making for them is critical. The reason is that multiple weights affect the combinatorial contribution of each sub-factor to calculate the overall value of each factor. Thus, one of the solutions is to apply a hierarchy methodology to cope with such a problem. AHP applies a hierarchy procedure to divide the problem into multiple levels of judgment procedures [7] and it is a widely used approach for decision making. In AHP the analysis processes have to be accomplished step by step based on our subjective judgment of each sub-factor. The main advantage of applying AHP before vulnerability quantification in Fuzzy Logic is that it suggests whether our subjective judgment of attributes is reasonable or not. In other words, it measures the final judgment value based on some predefined criteria; if the value does not follow the framework rule, then the subjective judgment should be revised and repeated until the ideal consistency is achieved. Other than inconsistency detection through the decision-making process, it has the capability of handling the problem of qualitative values of the attributes. The whole prioritization in AHP comprises five steps: (1) constructing the hierarchy structure of the main problem (2) constructing the pairwise comparison matrix based on the number of security sub-factors (3) calculating the relative weight of each sub-factor at each level (4) calculating the total weight of each sub-factor in each row and, (5) evaluate and check the consistency of judgment.

If there are *n* sub-factor for each factor, then there would be *n(n-1)/2* pair-wise comparison of sub-factors based on our predefined ratio scale. The ratio scale is reciprocal [7] and the comparison matrix *C* as in (1), is $n \times n$ where C_{ij} indicates the relative importance of *i* to *j*.

$$\text{Pairwise Elements:} \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \qquad (1)$$

where the sum of the values for each column of the pair-wise matrix is calculated as (2) [7][26]:

$$C_{ij} = \sum_{i=1}^{n} C_{ij} \qquad (2)$$

In order to achieve the normalized pair-wise matrix, each element in the matrix should be divided by its column total as shown in (3) [7][26].

$$X_{ij} = \frac{C_{ij}}{\sum_{i=1}^{n} C_{ij}} \qquad (3)$$

where X_{ij} indicates the normalized pair-wise matrix, and eventually, in order to generate the Weighted Matrix or Criteria Weight (CW) (4), the summation of the normalized column of the matrix should be divided by the number of sub-factors used as the criteria in AHP [7][26], where W_{ij} indicates weigh of each sub-factor.

$$CW_{ij} = \frac{\sum_{i=1}^{n} X_{ij}}{n} \qquad (4)$$

The final step is to determine whether the weighing processes for each sub-factor is consistent or not. Once the final matrix (CW) is generated we calculate the eigenvalue of matrix *M* with the corresponding eigenvector [26]. The consistency vector is calculated by multiplying the pair-wise matrix by the weight vector that is called Weighted Sum Value (WSV). The result of the average (WSV/CW) provides λ which is the eigenvalue. As shown in (5), the Consistency Index (CI) is calculated as follows:

$$\text{Consistency Index (CI)} = (\lambda - n)/n - 1 \qquad (5)$$

Satty [7] provided a mathematically-proven table to provide the Consistency Index (CI) for a randomly generated matrix that is called Random Index (RI). For example, a random index for a randomly generated 4×4 matrix is 0.89. Thus, to calculate the Consistency Ratio (CR) to determine if the matrix is consistent, CI should be divided by RI. If the result is less than 0.1 then the matrix is considered consistent which means there are no occurrences of combinations of pairs that bring about uncertainty, otherwise the entire decision making should be repeated with the different subjective judgment of attributes.

Since the illustration and implementation of all security sub-factors are beyond the scope of this paper we show only the implementation of “Runtime Security” as the main factor of Web Applications that is derived from Figure 2.

As shown in Table 2 we have four sub-factors in “Monitoring and Logging” that comprises “Firewall and Antivirus”, “XSS”, “Filtering Policy” and “Scanning and Pen Testing” that generates six pair-wise comparisons (in green) based on predefined scales that we obtained based on the analysis of each sub-factors’ role and contribution to the security of Web Applications. This table also addresses the normalized pair-wise matrix (in blue) based on previous steps (of green) with the division of each element of the columns with the corresponding SUM value of that column. The normalized pair-wise matrix is applied to generate Criteria Weights (CW) in Table 3.

Table 2. pair-wise comparison matrix of monitoring and logging

Pair-wise Matrix Comparison / Normalized pair-wise Matrix	Firewall and Antivirus	XSS	Filtering Policy	Scanning and Pen Testing
Firewall and Antivirus	1	5	4	3
	0.56	0.384	0.695	0.473
XSS	0.2	1	0.25	0.333
	0.112	0.076	0.043	0.052
Filtering Policy	0.25	4	1	2
	0.14	0.307	0.137	0.315
Scanning and Pen Testing	0.333	3	0.5	1
	0.186	0.23	0.086	0.157
Pair-wise comparison SUM	1.783	13	5.75	6.333

Table 3. criteria weights of monitoring and logging

	Firewall and Antivirus	XSS	Filtering Policy	Scanning and pen Testing
Criteria Weights (CW)	0.528	0.07	0.224	0.164

After calculating Weighted Sum Value (WSV), based on the aforementioned explanation, we measure the Consistency Ratio, (CR = CI/RI = 0.06479), which is less than 0.1 to prove that the matrix is reasonably consistent. We repeat all steps of CW calculation that we have accomplished for sub-factors of "Monitoring and Logging" to calculate the weights of sub-factors in "Encryption". The "Encryption" factor comprises two security sub-factors of "HTTP/HTTPS" and "Other Encryptions". After repeating the previous steps we obtained CW as follows in Table 4:

Table 4. criteria weights of encryption

Normalized pair-wise Matrix	HTTP/HTTPS	Other Encryptions	Criteria Weights
HTTP/HTTPS	0.666	1.333	0.999
Other Encryptions	0.333	0.666	0.499

When the process of weight calculations for all successors are fulfilled and the consistency of the matrix is assured, bottom-up aggregation is required to prioritize them as input variables in FIS.

This procedure will be addressed in section 3.4 to show how each sub-factors weight contributes to being mapped to linguistic variables in Fuzzy Logic.

3.4. Multi-layered Fuzzy Logic Implementation

One of the major problems in quantifying vulnerabilities of a system is that we are involved with linguistic variables (i.e., “very good”, “medium” or “very bad”) to describe the quality of security or vulnerability of that system. Moreover, there are no sophisticated and standard metrics to evaluate all available security sub-factors of a system that are linked to vulnerabilities. These two issues are, in part, inextricably interwoven in the context of system security, especially when we are considering vulnerability quantification. For that, we apply a Multi-layered Fuzzy Logic (MFL) to handle such intricate issues of vulnerability measurement.

Fuzzy Logic was proposed and implemented by Zadeh [5] in 1965. One of the essential components of Fuzzy Logic is the Membership Function (MF) which provides the degree of truth for measurement [27]. The MF’s value is always limited to the interval [0-1]. In Fuzzy Logic, there are four steps in measurement: (1) Fuzzification that generates a Membership value (2) Rules that use linguistic variables in an if-then format (3) Fuzzy Inference System (FIS) that formalizes if-then rules and maps input attributes to output space and, (4) Defuzzification that extracts the quantified values from FIS that is called the actual crisp number [27].

In the Fuzzification process, we use the Triangular Fuzzy model for MF [28] to evaluate the linguistic variables achieved from DOT’s security expert on the questionnaire. Two important properties of Fuzzy Logic are considered in the fuzzy rules in Fuzzification: (1) $\mu_{A\cap B}$ (x)=max[μ_A (x),μ_B (x)] | x∈X, and (2) $\mu_{A\cup B}$ (x)=min[μ_A (x),μ_B (x)] | x∈X. In Defuzzification process to generate the crisp number, we use Center of Area (COA) to calculate the area that MF covers within the range of the output variable as shown in (6) where x is the value of the linguistic variable, and x_{min} and x_{max} indicate the range of the linguistic variable and X is universal discourse [29].

$$\text{COA} = \frac{\int_{Xmin}^{xmax} f(x).x\,dx}{\int_{Xmin}^{xmax} f(x)\,dx} \qquad (6)$$

While multiple Fuzzy Logic applications in the related area simply employ fuzzy rules with the same weight as 1 when processing the variables, we put emphasis on considering each sub-factor as a completely specific influencer in Web Application security. As shown in Figure 4 the FIS for each security sub-factor of “Web Application” is designed separately in a backward manner to attain the parent node in each layer.

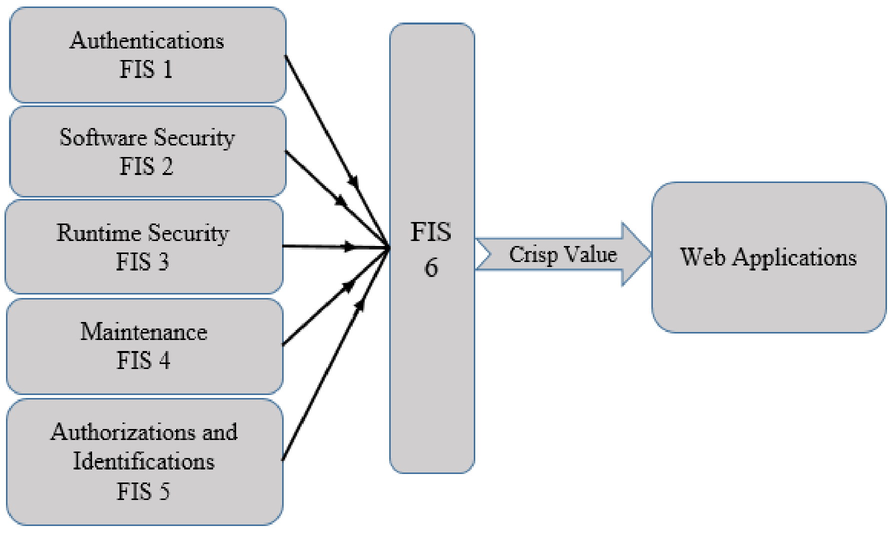


Figure 4. MFL for Web Applications in DOT

The results of the first round FIS (i.e., FIS 1 to FIS 5) will be obtained to contribute cumulatively to the second round FIS (i.e., FIS 6) to obtain the final crisp value for Web Applications. As shown in Figure 5 the backward aggregation of “Monitoring and Logging” and “Encryption” provides the value of “Runtime Security” as one of the main security factors of Web Applications in DOT. The process of vulnerability measurement in Fuzzy Logic, in this case, should take place in a backward flow from Figure 5, as the input, to Figure 4.

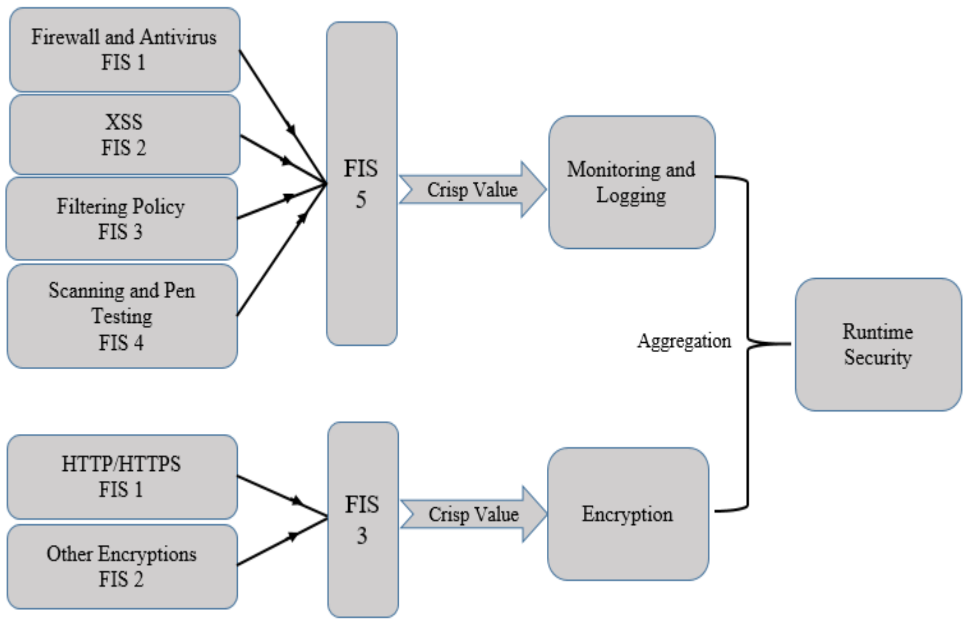


Figure 5. MFL for Runtime Security

After receiving the final result of the questionnaire from DOT’s security experts, we categorize the answers to some specific groups. The number of groups depends on the number of sub-factors and the role or the importance of each sub-factor in the security contribution of each level. The primary reason for this type of classification is that the answers of the questionnaire are in the format of “very low”, “low”, “medium”, “high” and “very high”. The weights in the interval of [0-10] are assigned to each sub-factor in the questionnaire. In other words, this way of linguistic-variable-description to evaluate the answers determines what fuzzy subset from “very low” to “very high” each sub-factor belongs to. As shown in Figure 6 all weights obtained from AHP are assigned to each sub-factor to contribute to the security of predecessors as shown in Figure 7. Furthermore, Figure 7 depicts the main factor “Runtime Security” that comprises four Groups: (1) two groups belong to “Monitoring and Logging” factor which is the result of interior FIS attributes such as “Firewall and Antivirus”, “XSS”, “Filtering Policy” and “Scanning and Pen Testing” (2) two groups belong to “Encryption” factor which is the result of interior FIS layer “Encryption” such as “HTTP/HTTPS” and “Other Encryptions”.

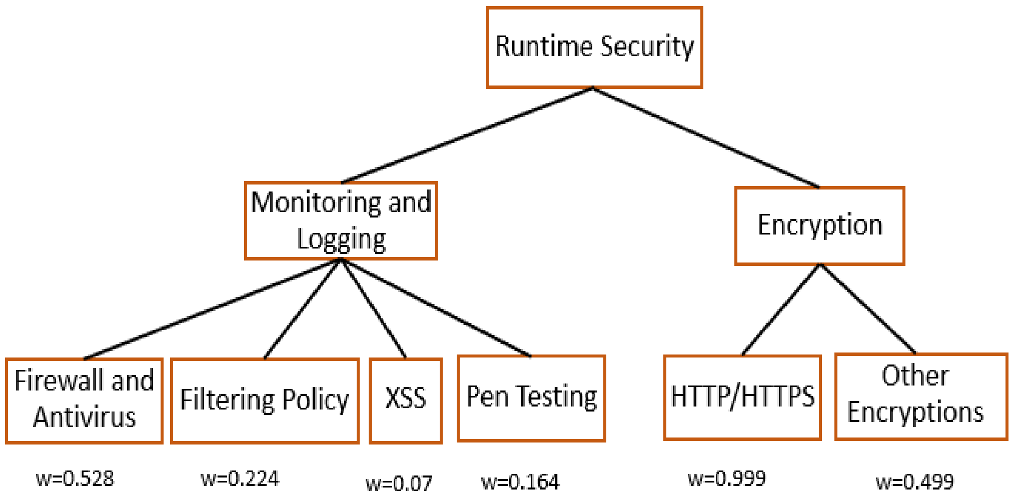


Figure 6. Assigned weights to each sub-factor based on AHP

As we mentioned before, researchers usually apply the Fuzzy Logic methodology such that the entire fuzzy rules are assigned with the same weights. However, we show that each rule with its specific weight of contribution in the security of a system would provide a more precise vulnerability measurement. As shown in Figure 7 each weight is assigned to the corresponding security sub-factor. If the number of sub-factors varies in two different main factors then, depending on the number of calculated weights, they are assigned to the fuzzy subsets. For instance, in “Monitoring and Logging” we weighed four security sub-factors which are simply assigned to each fuzzy subset, but in the “Encryption” factor each weight is assigned to two fuzzy subsets based on the value of the weights. The lower weights represent the lower fuzzy subsets and the higher ones take the higher subsets. The point in this method is that we always assign the weight with the highest value 1 to the fuzzy subset “very high”.

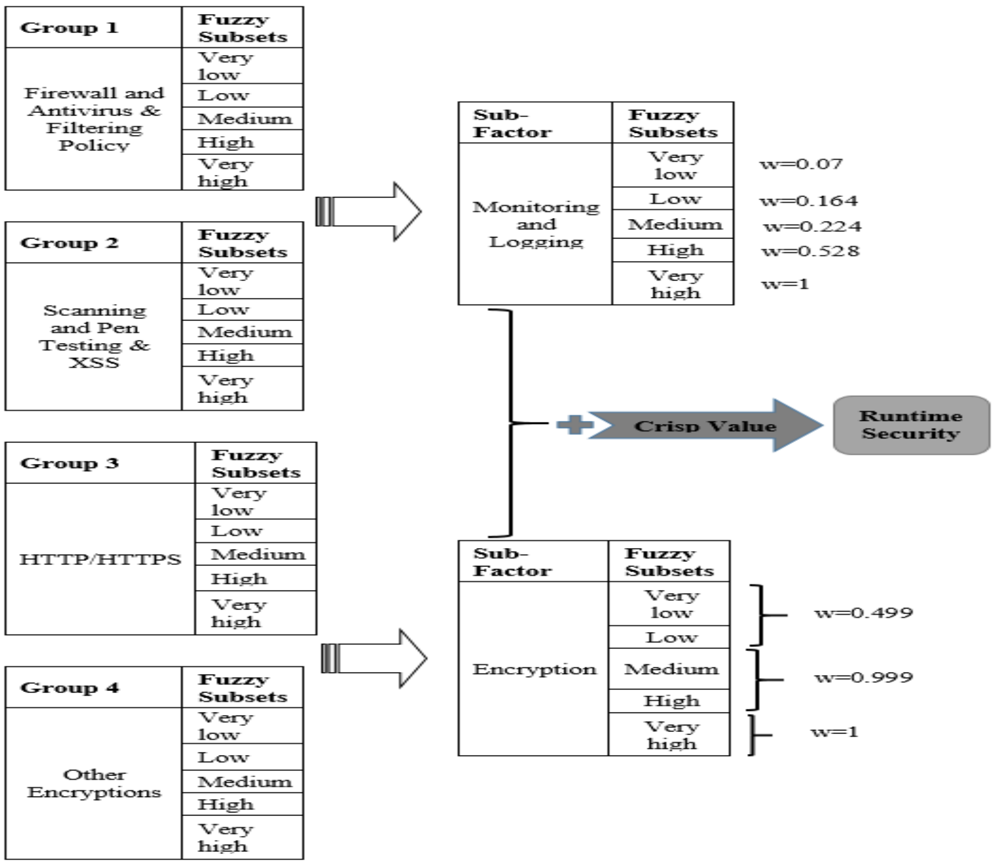


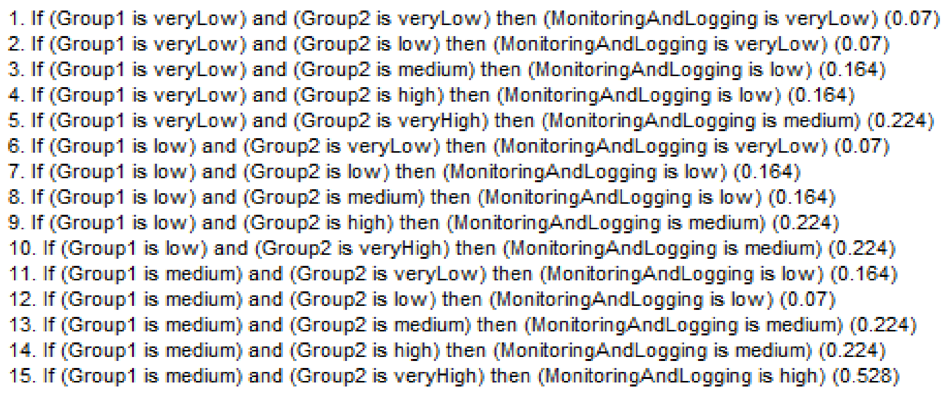
Figure 7. Fuzzy subset implementation for Runtime Security based on AHP

4. Implementation

The implementation of all security factors in the DOT's Web Applications is beyond the scope of this paper. We show only the implementation of "Runtime Security" to show the work procedure. We define the fuzzy if-then rules to map the inputs to the outputs in the MFL. This describes the conditions of the antecedent and consequent proposition. The advantage of applying fuzzy rules is that they work in parallel which does not affect the order or the sequence of the rules. Thus, we generate fuzzy rules based on the number of linguistic variables that describe the level of security in each question of the questionnaire. The more the number of linguistic variables, the more the number of rules. As shown in Figure 8 we defined 25 fuzzy rules derived from two groups in which each group has five linguistic variables.

These rules are designed for the first group based on the COA methodology. The weights are shown at the end of each line of the rule. Decision making based on a single fuzzy set occurs through aggregation to combine fuzzy sets from if-then rules to a single fuzzy set. For that, we use the max-min technique (7) for aggregation and the final value is achieved as follow:

$$\text{Final Value} = \max(\text{Group}_1, \text{Group}_2, \ldots, \text{Group}_N) \qquad (7)$$



1. If (Group1 is veryLow) and (Group2 is veryLow) then (MonitoringAndLogging is veryLow) (0.07)
2. If (Group1 is veryLow) and (Group2 is low) then (MonitoringAndLogging is veryLow) (0.07)
3. If (Group1 is veryLow) and (Group2 is medium) then (MonitoringAndLogging is low) (0.164)
4. If (Group1 is veryLow) and (Group2 is high) then (MonitoringAndLogging is low) (0.164)
5. If (Group1 is veryLow) and (Group2 is veryHigh) then (MonitoringAndLogging is medium) (0.224)
6. If (Group1 is low) and (Group2 is veryLow) then (MonitoringAndLogging is veryLow) (0.07)
7. If (Group1 is low) and (Group2 is low) then (MonitoringAndLogging is low) (0.164)
8. If (Group1 is low) and (Group2 is medium) then (MonitoringAndLogging is low) (0.164)
9. If (Group1 is low) and (Group2 is high) then (MonitoringAndLogging is medium) (0.224)
10. If (Group1 is low) and (Group2 is veryHigh) then (MonitoringAndLogging is medium) (0.224)
11. If (Group1 is medium) and (Group2 is veryLow) then (MonitoringAndLogging is low) (0.164)
12. If (Group1 is medium) and (Group2 is low) then (MonitoringAndLogging is low) (0.07)
13. If (Group1 is medium) and (Group2 is medium) then (MonitoringAndLogging is medium) (0.224)
14. If (Group1 is medium) and (Group2 is high) then (MonitoringAndLogging is medium) (0.224)
15. If (Group1 is medium) and (Group2 is veryHigh) then (MonitoringAndLogging is high) (0.528)

Figure 8. Fuzzy rules for Runtime Security based on AHP

The three-dimensional plot in Figure 9 maps group 1 and group 2 to the "Monitoring and Logging" security factor in Web Applications. A range from 0 to 9 is presented in the vertical axis. The value 0 indicates the least security grading (maximum vulnerability) and the value 9 indicates the most security set-out (least vulnerability) accordingly.

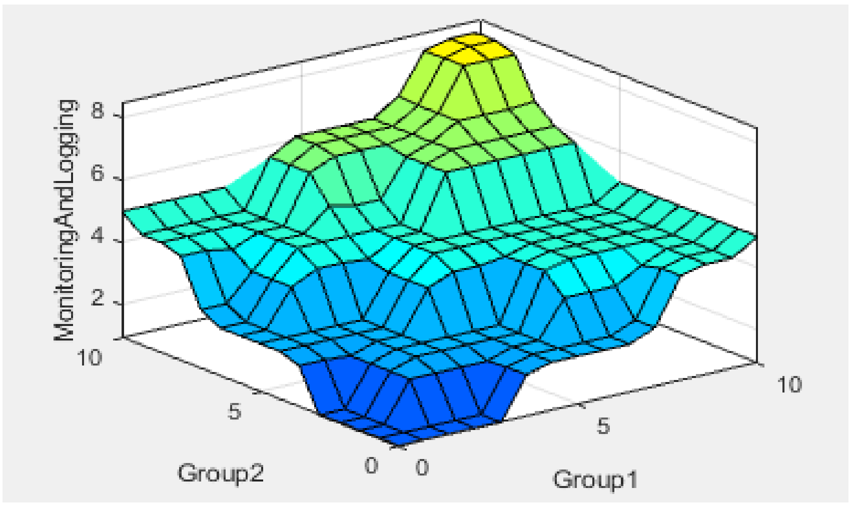


Figure 9. Output curve for Runtime Security based on AHP

As shown in figure 9 there is a smooth output (not flat plateaus) which indicates the general incremental trend illustrating that we do not have many input combinations that produce the same output. This figure shows the maximum value of "Monitoring and Logging" happens at the peak value of the intersection of Group1 and Group2. This is the best evidence to show how the vulnerability measurements in different inputs are distinguished.

We applied MFL to all aforementioned four groups of security factors derived from Figure 7 to quantify vulnerabilities of them based on the weights that prioritize each sub-factor in AHP. In each measurement process, we have two conditions for comparison: (1) uneven weights acquired from AHP are assigned to fuzzy rules and, (2) weights with the same value 1 are assigned to all fuzzy rules. As shown in Figure 10, fuzzy inference processes are employed to facilitate the process of interacting with the input value to obtain the output value based on aggregation in each fuzzy set for "Monitoring and Logging". The columns in the left and the middle indicate the if-part fuzzy rule which is known as "antecedent" and the last column in the right represents the then-part which is called “consequent”. The average input and output are shown on top of each column. In this case, Group 1 has an average value of 3.99, and Group 2 has an average value of 5.5 out of 10. The result value for “Monitoring and Logging’ is 4.48 out of 10. This is calculated at the last plot of the third column (at the very right bottom) which is based on the aggregation of the weighted decisions in the system from the maximum value of all minimum inputs captured in COA. The important point here is that the value 4.48 indicates the security of “Monitoring and Logging” not the vulnerability. In order to obtain its vulnerability, we have to subtract it from 10 (that is 10 - 4.48=5.52).

As shown in Figure 11 the MFL is applied to the same groups as Figure 10 but in this measurement process, all fuzzy rules have the same weight 1. The output for the security of “Monitoring and Logging” is 4.46 that quantify the vulnerability with the value 5.54.

We repeat the same measurement processes for the “Encryption” factor. Figure 12 shows that the security value in Groups 3 and 4 (based on the aggregation of the weighted decision) has the value 6.54, which quantifies its vulnerability by the value 3.46 out of 10. On the other hand, Figure 13 addresses that the security and vulnerability values of Groups 3 and 4 (while all fuzzy rules are assigned with the same weight 1) are 6.47 and 3.53 respectively. Ultimately, in table V we calculated the total quantified vulnerability value of “Runtime Security” in both conditions of “uneven Weights” and “Weight 1” to show the differences in measurement values.

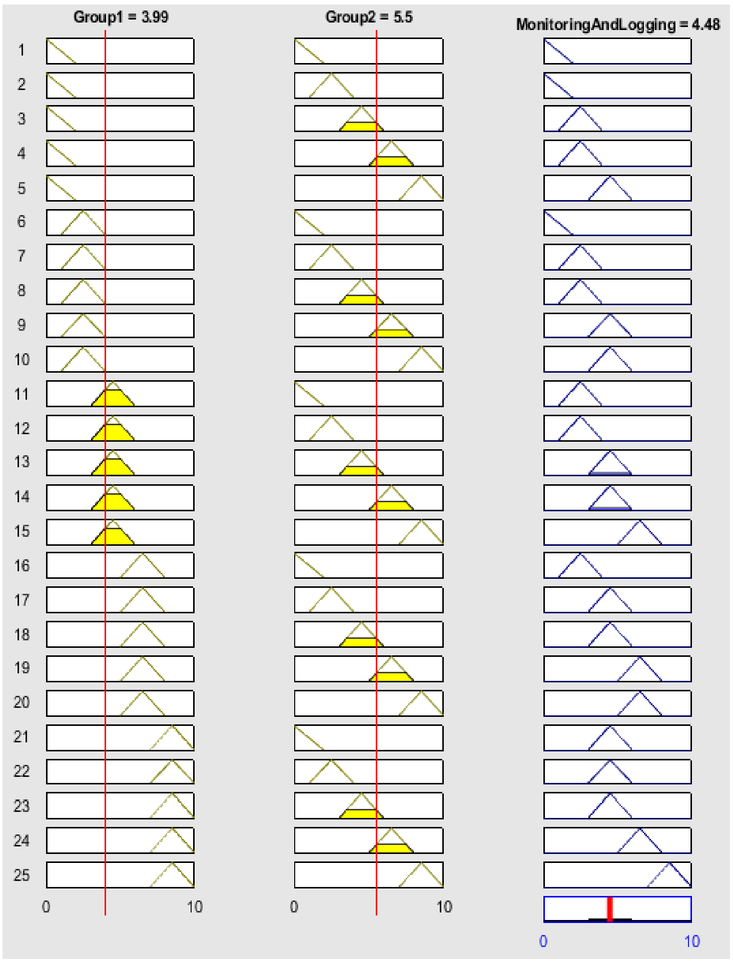


Figure 10. Fuzzy inference processes for Monitoring and Logging based on AHP

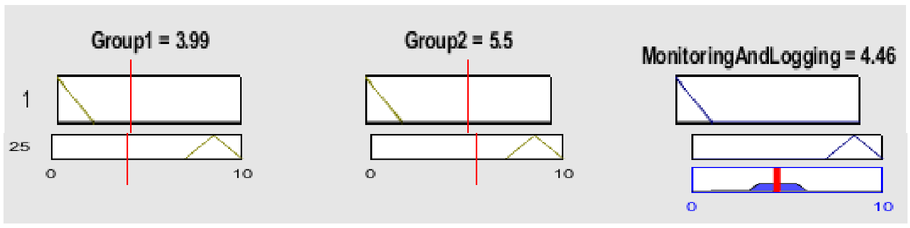


Figure 11. Fuzzy inference processes for Monitoring and Logging with the same weight 1

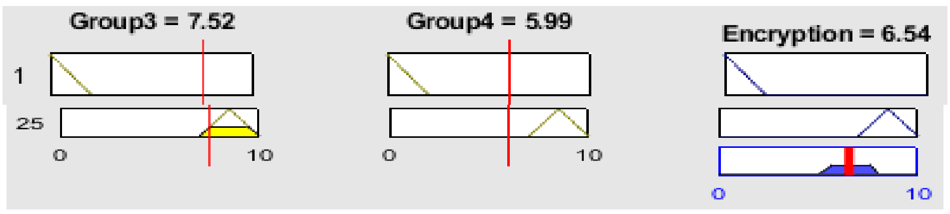


Figure 12. Fuzzy inference processes for Encryption based on AHP

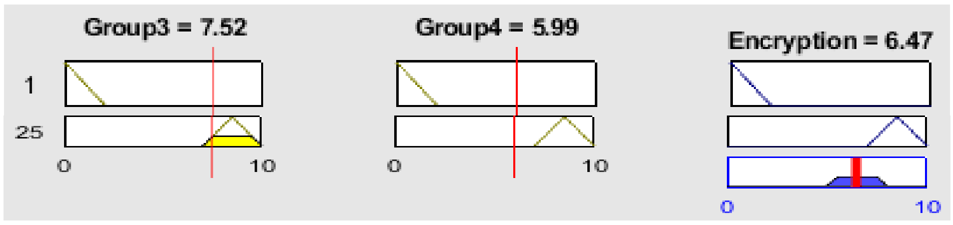


Figure 13. Fuzzy inference processes for Encryption with the same weight 1

Table 5. criteria weights of encryption

Runtime Security	**Monitoring and Logging**	**Encryption**	**Total (Avg)**
Weight 1	5.54	3.53	4.53
Uneven Weights	5.52	3.46	4.49

Since we are using the Mamdani type of rules as defined earlier, $R_1 =$ If x_1 is A_{i1} and … and x_{iN} is A_{iN} then y is B_1, considering that we may have redundant rules or variables in the process of rule construction, in the worst-case scenario to measure the vulnerability of a system, there are a finite number of variables that propose super-polynomial time or exponential complexity. If the model contains x variables and maximum r fuzzy terms (here we do not have more than five) then the order is $O(r^x)$ based on the search for the rules for specific input. (in our case it is considered as polynomial-time).

5. Conclusions and Future Work

The lack of a comprehensive approach to vulnerability quantification is an unsolved issue among cybersecurity researchers. The main reason for this is because the quality description of a system's security is usually in the format of human linguistic variables, and this model of security expression always results in multiple interpretations from the experts. In this paper, we proposed a Multi-layered Fuzzy Logic (MFL) approach to measure the vulnerability of Web

Applications in DOT based on the prioritization of security sub-factors from AHP. The role and the contribution of each security sub-factor are extracted based on the answers we receive from security experts of DOT in a questionnaire. We quantified the vulnerability of "Runtime Security" in two different scenarios: (1) Uneven weights for fuzzy rules and, (2) Use the same (even) weight 1 for all fuzzy rules. Since the measurement in the former one is based on the exact contribution of each security sub-factor in the security of Web Applications, it conveys a more precise weight while quantifying the vulnerability than the latter. The findings show that when we quantify the vulnerabilities based on even weight 1 for all rules, it overestimates the vulnerability of the system with a value of 0.04 out of 10. This value may not represent a significant difference in vulnerability measurements of the two scenarios in this scale. However, in a large corporation, this difference would be conspicuous. In other words, although both methodologies calculate the vulnerability of the system well (very precisely), we prefer to calculate each metric based on the AHP method to conduct the assessment based on the exact weight of each security sub-factor, because this method does not overestimate the security of the system. To our knowledge, this is the first methodology to quantify vulnerabilities based on GQM, security standards, and AHP in Fuzzy Logic.

To recapitulate briefly, the proposed methodology with the findings in this paper, regardless of weighting the security sub-factors or not, is a novel approach to quantify the vulnerability of not only Web Applications, but all components of network security.

This methodology is very efficient, flexible to add layers, tolerant to imprecise data, and easy to understand. The primary limitation is that the processes of rule construction largely increase depending on the number of variables.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] "Announcements," IBM News Room. [Online]. Available: https://newsroom.ibm.com/2019-07-23-IBM-Study-Shows-Data-Breach-Costs-on-the-Rise-Financial-Impact-Felt-for-Years.

[2] "Acunetix Web Application Vulnerability Report 2019," Acunetix. [Online]. Available: https://www.acunetix.com/acunetix-web-application-vulnerability-report.

[3] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing," *2010 IEEE Symposium on Security and Privacy, 2010.*

[4] A. Razzaq, K. Latif, H. F. Ahmad, A. Hur, Z. Anwar, and P. C. Bloodsworth, "Semantic security against web application attacks," *Information Sciences*, vol. 254, pp. 19–38, 2014.

[5] L. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338–353, 1965.

[6] V. R. Basili and S. Green, "Software Process Evolution at the SEL," *Foundations of Empirical Software Engineering*, pp. 142–154.

[7] T. L. Saaty, "How to Make a Decision: The Analytic Hierarchy Process," Interfaces, vol. 24, no. 6, pp. 19–43, 1994.

[8] P. J. C. Nunes, J. Fonseca, and M. Vieira, "phpSAFE: A Security Analysis Tool for OOP Web Application Plugins," *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015.*

[9] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, "W-VST: A Testbed for Evaluating Web Vulnerability Scanner," *2014 14th International Conference on Quality Software, 2014.*

[10] B. Eshete, A. Villafiorita, K. Weldemariam, and M. Zulkernine, "Confeagle: Automated Analysis of Configuration Vulnerabilities in Web Applications," *2013 IEEE 7th International Conference on Software Security and Reliability, 2013.*

[11] M. K. Gupta, M. C. Govil, and G. Singh, "Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications," 2015 *12th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2015.*

[12] W. Ze, "Design and Implementation of Core Modules of WEB Application Vulnerability Detection Model," 2019 *11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 2019.*

[13] J. Wal, M. Doyle, G. A. Welch, and M. Whelan, "Security of open source web applications," 2009 *3rd International Symposium on Empirical Software Engineering and Measurement, 2009.*

[14] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," *ACM SIGPLAN Notices*, vol. 41, no. 1, pp. 372–382, Dec. 2006.

[15] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, "Toward Automated Detection of Logic Vulnerabilities in Web Applications," *USENIX Security'10: Proceedings of the 19th USENIX Conference on Security, Aug. 2010.*

[16] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: a static analysis tool for detecting Web application vulnerabilities," 2006 *IEEE Symposium on Security and Privacy (S&P06), 2006.*

[17] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring," *Proceedings of the twelfth international conference on World Wide Web - WWW 03, 2003.*

[18] F. Ö. Sönmez, "Security Qualitative Metrics for Open Web Application Security Project Compliance," *Procedia Computer Science*, vol. 151, pp. 998–1003, 2019.

[19] P. R. L., L. C. S., D. Jagli, and A. Joy, "Rational Unified Treatment for Web application Vulnerability Assessment," 2014 *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014.*

[20] K. Philipsen, "Web Application Vulnerability Assessment Discovering and Mitigating Security Issues in Web Applications," *Cybertrust.*

[21] A. Gray and S. Macdonell, "GQM A Full Life Cycle Framework for the Development and Implementation of Software Metric Programs (1997*)," Implementation of Software Metric Programs" , Australian Software Measurement Conference*, pp. 22–35.

[22] J. C. Abib and T. G. Kirner, "A GQM-based tool to support the development of software quality measurement plans," *ACM SIGSOFT Software Engineering Notes*, vol. 24, no. 4, pp. 75–80, Jan. 1999.

[23] A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti, G. Oldano, and E. Orazi, "Applying GQM in an industrial software factory," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 7, no. 4, pp. 411–448, 1998.

[24] V. Basili, J. Heidrich, M. Lindvall, J. Munch, M. Regardie, and A. Trendowicz, "GQM Strategies -- Aligning Business Strategies with Software Measurement*," First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2007.*

[25] M. Shojaeshafiei, L. Etzkorn, and M. Anderson, "Cybersecurity Framework Requirements to Quantify Vulnerabilities Based on GQM," *SpringerLink*, 04-Jun-2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-31239-8_20.

[26] J. Watkins and L. Ghan, "AHP Version 5. 1 users manual. [Analytical Hierarchy Process (AHP)]," Jan. 1992.

[27] H.-J. Zimmermann, "Fuzzy Sets, Decision Making, and Expert Systems," 1987.

[28] W. Pedrycz, "Why triangular membership functions?," Fuzzy Sets and Systems, vol. 64, no. 1, pp. 21–30, 1994.

[29] E. Ngai and F. Wat, "Fuzzy decision support system for risk analysis in e-commerce development," *Decision Support Systems*, vol. 40, no. 2, pp. 235–255, 2005.

[30] R. Luh, M. Temper, S. Tjoa, S. Schrittwieser, and H. Janicke, "PenQuest: a gamified attacker/defender meta model for cyber security assessment and education," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 1, pp. 19–61, 2019.

[31] A. Shafee, M. Baza, D. A. Talbert, M. M. Fouda, M. Nabil, and M. Mahmoud, "Mimic Learning to Generate a Shareable Network Intrusion Detection Model," *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020.