Assuring Quality in Vulnerability Reports for Security Risk Analysis

Deepak Subramanian, Ha Thanh, Le and Peter, Kok Keong, Loh

School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
Email: DEEP0018@ntu.edu.sg

***Abstract*-Web application scanners detect and provide some diagnoses for specific vulnerabilities. However, scanner performance as well as the damage potential of different vulnerabilities varies. This undermines the development of effective remediation solutions and the reliable sharing of vulnerability information. This paper describes the development of fuzzy classification metrics that are used to grade web application scanners and vulnerabilities so that scanner performance can be evaluated and confidence levels can be computed for vulnerability reports. These metrics help derive a level of assurance that will support security management decisions as well as enhance effective remediation efforts.**

Keywords
Fuzzy classifiers, confidence level, calibration, scanner, vulnerability, web application

I. Background and Motivation

Contemporary detection of software vulnerabilities in web-based systems is accomplished via web application scanners [21, 25]. However, depending on the capabilities of these scanners, different vulnerability reports generated will have a widely varying level of trustworthiness. This raises critical concerns especially when these reports are used to estimate system risks for management decisions and the development of remediation processes.

Risk analysis is inherently a complex process fraught with ambiguity and uncertainty. Traditional risk approaches are usually based on assumptions of known vulnerabilities or threats and are thus not suitable for contemporary web services and applications that exhibit a degree of platform inter-operability and dynamic content. In different web applications, some vulnerabilities are also more dangerous than others in terms of potential damage/risks [1, ICIMP 2009]. These issues create a challenge to develop a quality assurance mechanism for scanner generated reports. Qualified reports can then support a trusted level of analysis of system risk as well as being a more dependable resource of shared system security information.

Our assurance mechanism described in this paper focuses on supporting more reliable risk analysis of web-based systems and is based on fuzzy metrics that are used to calibrate scanner performance as well as vulnerabilities. Our approach also forms part of a system framework that achieves standardization of scanner reports across different web technologies [2-4]. The standardization is necessary since there has been a rise in the number and type of scanners and vulnerabilities. Scanner algorithms evolve as the vulnerabilities evolve. For any organization that had a greater requirement for security, it would be more advisable to rely on several algorithms instead of just one since these algorithms perform differently in different scenarios. The scanner results can then be collated to give a more significant output in a standardized format. Our approach would benefit the Security Administration and Audit groups of an organization or enterprise in ensuring scalable security enforcement and compliance against an unpredictable vulnerability backdrop.

In this paper, our approach utilizes the standardized scanner results generated, together with scanner performance and vulnerability calibrations, to compute associated confidence levels with these results.

The rest of the paper is organized as follows. Section II presents a review of existing research. Section III sums up the research issues based on the review and describes the requirements. Section IV details the design of the quality assurance metrics while section V presents the design of the scanner and vulnerability grading systems. Section VI exemplifies the calculation of the 1st and 2nd degree confidence levels for vulnerability reports while Section VII provides an illustrative example to describe the working of the framework. Section VIII concludes the paper followed by the references.

II. Review of Existing Research

It is difficult for decision makers to identify entire network threats and collect precise and adequate data to estimate all probable risks due to vulnerabilities or threats. Furthermore, risk analysis for web service security and applications is not only limited to determining recognized web threats, but should also estimate potential risks. A

review of the more recent works that have had some influence in this research is described in this section.

In [14] an extended form of the *Pseudo-Order Preference Model (POPM)* was used to estimate the imprecise risk of web services based on richness of information and to determine their ranking using a weighted additive rule. A fuzzy logic based approach was used to calculate information characteristics provided by the web service. There are 3 models used in this process a) Pseudo-order preference model, b) Semi-order preference model and c) Complete-preorder preference model. Each model is executed if and only if a condition is reached. Each model was given an Outranking relation which the research states would affect the decision making capabilities for the risk analysis. The decision makers have been stated as useful parameters in helping the experts making their decisions. The future work in this model includes the selection of an appropriate threshold for the preference relation, defining an appropriate threshold for the Indifference preference. The use of fuzzy logic by this research, however, does not extend to measure differing security tool performance and vulnerability severity.

In the research [15], the software code has been taken and analyzed for security patterns. The paper shows the results of experimenting with J2EE code with a MySQL back-end and JBoss Application Server. The various software security patterns of Intercepting Validator, Guard of Secure Proxy with Secure Pipe, Container Managed Security and Secure Logger have been analyzed with how implementation of each affects the vulnerability being used on the system. A fuzzy approach has been used by having a linguistic value for every generated fuzzy equivalent range such as low, medium, high, very high etc., This has been used to analyze the effectiveness of the various security patterns. The effectiveness of the patterns against primary attack evens i.e. events that lead to execution of an attack has also been analyzed. While each pattern has a varying effectiveness in varying scenarios, the code with security patterns implemented has been proven to always be more secure to the ones that are not following the patterns. Future work of the project involves the creation of newer patterns that have not been mentioned above. This approach uses a whitebox methodology to test for code patterns that have a less likelihood of getting affected by certain vulnerabilities. Our approach compliments this research by ensuring if these code patterns have been designed and implemented securely.

The research [16] describes the need to prevent or manage the damage caused by security threats. The research describes that the web-server based applications must be made in such a way that they incorporate the ability to self-heal after an attack. The authors describe how this can be made possible by the basis of data obtained from anomaly detection. The anomaly detection data is then processed using a Discrete Finite Automate (DFA) to detect malicious web requests. An anomaly based detection combined with DFA needs to be trained in the beginning to find which anomaly detection data matches a true positive attack and the patterns of such requests are observed by the training algorithm. The patterns are then detected after the training and such requests are suggested to be blocked or sent to a more secure, but less functional server to protect the found and restored. This approach is suitable for systems that are holding highly critical data that cannot be changed but would need extensive training and validation and could be expensive to implement. It needs to be implemented at every server. It is not a preventive technique but a criterion for recovery when an attack is observed. Our approach, on the other hand, is a part of a framework that would be able to provide remediation based on the observed attack but not an automated recovery.

The research in [12] proposed a method for identifying and charting software exposure to un-patched vulnerabilities. Disclosed vulnerabilities are divided into 2 types. The first comprises of vulnerabilities that are publicly known with no patch available from the vendor. The second comprises of vulnerabilities that are publicly known with a patch available from the vendor. By calculating the Daily Vulnerability Exposure (DVE) for all un-patched vulnerabilities for a continuous period of time, an exposure chart is obtained. Using the chart's help, it is possible to ascertain how long a vendor takes to patch and if the patch is effective, by calculating the DVE after the patch date. The exposure chart could also be used to calculate the severity metrics used by the National Vulnerability Database (NVD). The DVE is a severity metric that is based on how much the vulnerability is graded in terms of time elapsed, from the date it is discovered till the date a patch is available. The vulnerabilities handled here are generally in the new and Relatively new categories of our approach. These have been described in the section V.

The research in [13] is oriented towards the quantitative characterization of the vulnerabilities in operating systems. A time-based model for the total vulnerabilities discovered is proposed and fitted to the data for Windows 98 and Windows NT 4.0. Being a time-based model, it is able to obtain an indication of the expectancy of the vulnerability being targeted based on the phase the system is in. An alternative effort-based vulnerability model analogous to software reliability growth models was also proposed. Both models fit well and the fit is significant, however, further development is necessary before confidence levels associated with the detection of vulnerabilities can be assessed.

III. Research Issues and Requirements

While data classification can act as an enabler for a more effective diagnosis and calculation of DVE helps in determining patch effectiveness [12], the widely varying detection capabilities encountered during scanning as well as the differing threat / risk levels posed by individual vulnerabilities have not been addressed.

The approaches [10][5][11] have been influential in validating the use of fuzzy logic in classification. The use of a neural network is an effective method in developing an inference engine. However, it needs to a lot of training data and this data can influence the working in a very significant way. By using suitable heuristics instead gives more stability to the system against any misclassification errors and also reduces any complexity that could be faced while training a neural network.

The scanner output could be right or wrong depending upon the algorithms used by the scanner and the database supported by the scanner. From the intrusion detection research stated above [5][11] it can be observed that the intrusion detection models also use several tools to first identify the various suspicious events and then have some decision making processes to deal with such observed events based on fuzzy diagnosis. This research also adopts such an approach to deal with the web vulnerabilities discovered by the scanners. However, this is the only similarity between the approaches in this research and the above stated works [10][5][11].

In our research, we address the variable detection capability of scanners and different threat / risk levels posed by individual vulnerabilities. Our approach grades web vulnerabilities and scanners quantitatively via expressions based on fuzzy truth values. The requirements of our approach are stated as follows:

1. Metrics for vulnerabilities and scanner performance may be calibrated empirically prior to analysis making this a more practical and flexible methodology.

2. Web vulnerabilities and scanner performance may be classified and ranked in a reliable and informative way.

3. Scales easily to cover new vulnerabilities, vulnerability variants and scanners.

4. Supports more effective management and remediation decisions and facilitates occurrence estimation of classified vulnerability.

IV. Preliminaries

In this section, we define and explain the terminologies that will be used in the rest of the paper.

Scanners are applications that use suitable algorithms to detect web vulnerabilities.

Fuzzy sets are sets whose elements have degrees of membership. An element mapping to the value 0 means that the member is not included in the fuzzy set, while a mapping to the universe of disclose, where the universe of disclose represents the entire set of members possible and the fuzzy sets they belong to. A diagrammatic representation is given in Figure 1. A value of 1 describes a fully included member. Mapped values strictly between 0 and 1 characterize the fuzzy members.

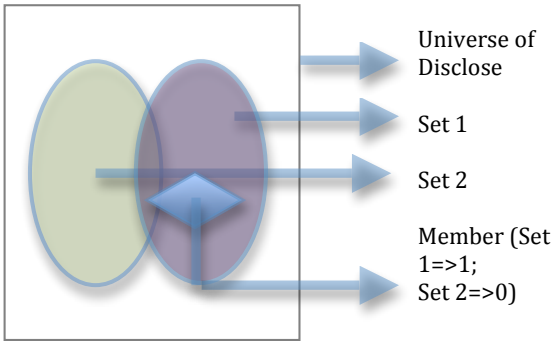

Figure 1. Venn Diagram of a fuzzy set

A *positive vulnerability* refers to a vulnerability that is present in the website at a specified instant and there is evidence to support it.

A *negative vulnerability* refers to a vulnerability that is not present in the website and can be proved to a satisfactory level.

The *calibration phase* is the time period during which fuzzy metrics are calibrated before the scanner is ready to generate vulnerability reports.

An *instance* of a vulnerability detected present (absent) by a scanner for a given website can be defined as that manifestation (or non-manifestation) of the vulnerability that occurs during a specified period of time where there has been no change in scanner algorithm, scanned website or vulnerability definition.

Test websites are those websites that have been custom designed to contain or not contain specified vulnerabilities for the purpose of testing the scanners. The test websites are used mainly in the calibration phase.

Ground truth is the true value of whether the vulnerability is present or absent. It is an absolute value i.e. the vulnerability is present or it is absent.

The *Likelihood Ratio* is the ratio of the probability that a particular vulnerability would be predicted when it matches the ground truth to the probability that it would be predicted erroneously.

Sensitivity is the proportion of correct detections of vulnerability presence out of all true instances of a particular scanner's detection. It can be computed in both a vulnerability specific way as well as in a scanner specific way. When calculated in a scanner specific way, it is averaged over all the vulnerabilities. It corresponds to the correct detection rate relative to ground truth.

Specificity is the proportion of false detections of vulnerability presence out of all false instances of a particular scanner's detection. It can be computed in both a vulnerability specific way as well as in a scanner specific way. When computed in a scanner specific way, it is averaged over all specified vulnerabilities.

Cross-site request forgery (CSRF) is an attack which forces an end user to execute unwanted actions on a web application in which the end user is currently authenticated.

Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

V. Design of Classification Metrics

In our proposed design (Figure 2), *calibration* forms an important and integral part of the framework. The calibration process makes use of two grading systems: *scanner grading system* and *vulnerability grading system*. Scanner and vulnerability metrics are first calibrated by the respective grading systems before any confidence levels and diagnostics are computed.

Grading can increase the reliability of reports obtained by allowing for their evaluation based on the grades of the various scanners that detected the specified vulnerability and threat/risk posed by the vulnerability. Grading of scanners and vulnerabilities are computed based on *scanner specific truth-values* and *vulnerability specific truth-values*, respectively. Scanner specific and vulnerability specific truth-values form fuzzy sets. The assurance of a scanner based on the vulnerability it is able to detect can provide an assurance of quality in the vulnerability reports provided by various scanners.

Using the grading results, low performance scanners can be selectively upgraded or omitted and vulnerabilities with high damage potential can be identified and affected systems isolated. Additionally, computation of report confidence can also be carried out. For example, a low confidence level obtained while the vulnerability is detected would mean that there is a low likelihood that the vulnerability is actually present. On the other hand, a high confidence level implies that there is a high probability that the detected vulnerability will not be a false positive. The confidence level thus obtained is an assurance of the risk analysis for the various vulnerabilities that has been done on the particular website location.

In the next few sub-sections, we detail the development of the framework design based on the requirements and from the assertions made. The assertions form an integral part of the system that provide a basis for the subsequently proposed metrics.

A. Assertions made

Assertion 1:
Some vulnerabilities are more difficult to exploit than others.

Not all vulnerabilities are equally susceptible to exploitation and the potential damage that can be caused will also not be the same. Hence, a *vulnerability grade system* needs to be present to provide a better diagnosis of the various vulnerabilities that are detected by the web application scanners.

Assertion 2:
Web-based vulnerabilities can be classified into 4 types, namely:

i. Evolved Vulnerability:
If there is recorded detection for the vulnerability and there also exists at least one recorded exploitation method that is still usable.

ii. Dormant Vulnerability:
If all recorded exploitation methodologies can no longer be used and there is at least one recorded exploitation method.

iii. Relatively-new Vulnerability:
If there is no recorded detection but there exists at least one recorded exploitation method.

iv. New vulnerability:
If there is no recorded detection or exploitation method for the vulnerability.

By classifying vulnerabilities into specific types, it is possible to evaluate the capability of scanners as well in a better way. For example, a less sophisticated scanner would not be expected to find a new or even relatively-new vulnerability. This would provide a credibility rating for the scanner with specific scanner metrics defined in the sections to follow. The severity of the vulnerability can also be ascertained to a certain degree with this approach. For example, a dormant vulnerability resulting from a series of successful patches will have a lower severity than an evolved one.

Assertion 3:
The difficulty of detection of an evolved vulnerability is directly proportional to the difficulty of exploiting it.

The above assertion is influenced intuitively by the notion that if a complex algorithm and/or extended process were needed to detect the vulnerability, a proportionate effort would be required in effectively exploiting it. In other words, if the vulnerability can be easily detected or observable then the skill level / effort needed for exploitation is correspondingly less. Given the above assertions and the fact that not all scanners will be able to deal with a particular vulnerability with the same degree of effectiveness, scanning capability must be graded. The capability of a scanner to effectively detect a vulnerability is represented by an index allocated to it known as S_{GRADE} (*Scanner Grade*). A scanner with a higher S_{GRADE} is then better suited to detect the vulnerability than one with a lower S_{GRADE}.

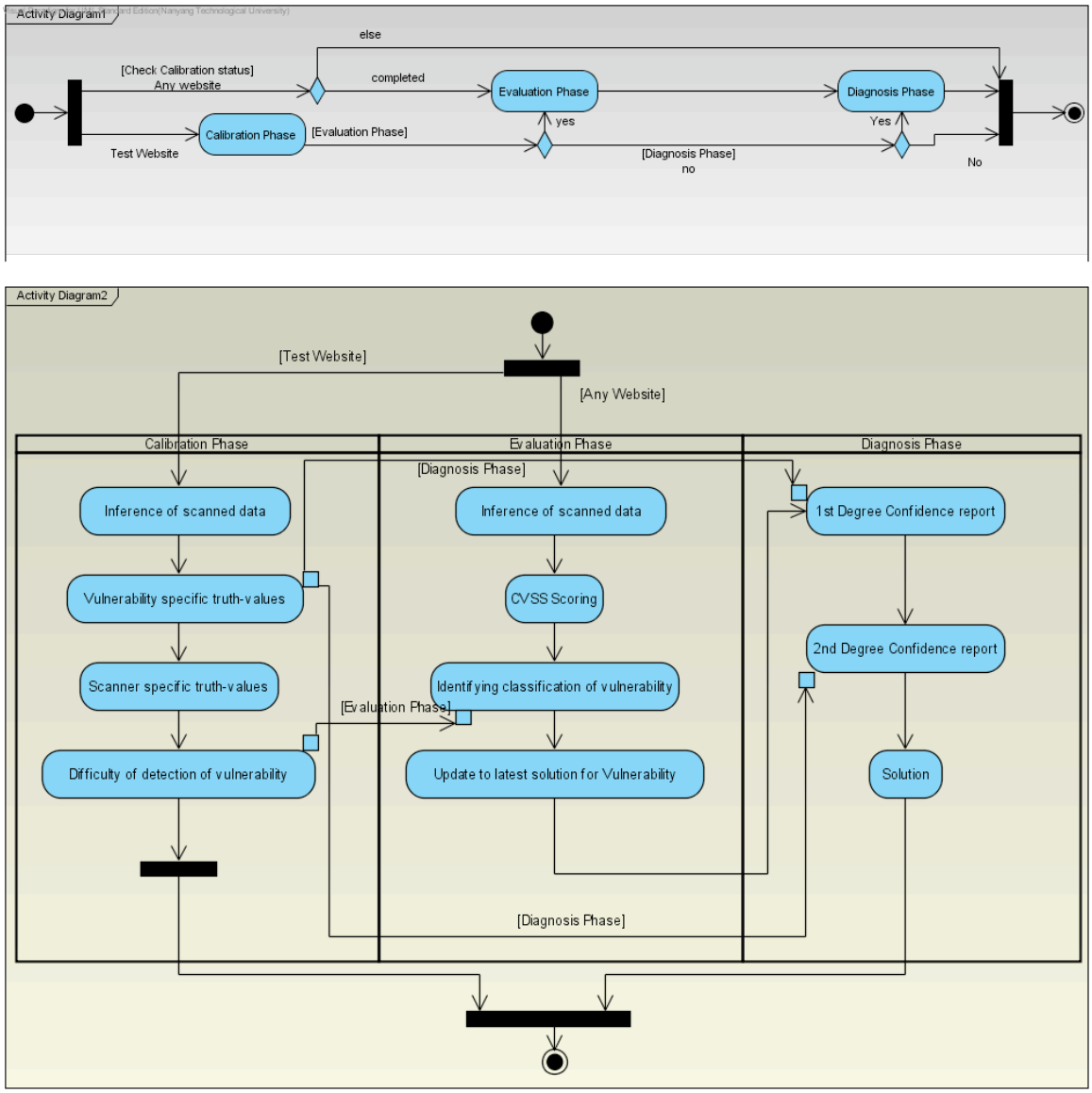

Figure 2: Overview of Framework Design

Assertion 4:
The impact level of a vulnerability (vulnerability-exploitability measure) is likely to vary with varying instances of detection and at varying locations.

It has been stated in assertion 1 as to how some vulnerabilities may be more difficult to exploit than others. It is also true that the same vulnerability may be present in a more exploitable location than others. Consider cross-site scripting at a site where the vulnerability may be exploitable directly. At a later time, perhaps after changes in site architecture, the same vulnerability may be exploitable only after a series of authorization pages. Hence, the exploitability of the cross-site scripting has changed for the same website at different instances of detection. Similarly, the exploitability of cross-site scripting may also vary due to platform differences at various sites.

Assertion 5:
Existence of one vulnerability may influence the prevalence of another.

Steps are usually taken to prevent some vulnerabilities in a system. Some other vulnerabilities may, however, still exist. These other vulnerabilities may directly or indirectly enable the avoided vulnerabilities to bypass the previous prevention schemes. An illustrative example is that of *cross-site request forgery* (CSRF) and *cross-site scripting* (XSS) [19]. If cross-site request forgery has been avoided by non-usage of JavaScript and secret tokens at each level (which is quite an effective methodology), the website is virtually protected from CSRF and typical scanners will also declare the same. However, if XSS has not been avoided, it can be used to get the tokens ahead of time and a hybrid use of CSRF can be realized which cannot be detected by the scanners.

Hence, we may define $R_{V1}(V2)$, where $R_{V1}(V2)$ is the likelihood of occurrence of vulnerability V1 when vulnerability V2 has occurred.

B. Vulnerability Specific Truth-Values

The various vulnerabilities have their own levels of difficulty as defined by the assertion 3. It is therefore a necessity to ascertain how scanners react to the various vulnerabilities. The scanners themselves need to be graded as well, which is described in the section V.C. After the computation of the vulnerability specific truth values and scanner specific truth values, the difficulty of detection of the vulnerability can be ascertained. The difficulty of detection is a useful estimation that can help in ascertaining the importance of detection thus ensuring a quality-based analysis of scanner detections.

Let the vulnerability specific truth values for vulnerability *j* be represented by $\{V(j)_{TP}, V(j)_{TN}, V(j)_{FP}\& V(j)_{FN}\}$. These are also known as {True Positive, True Negative, False Positive and False Negative}, respectively. These truth values can be used to derive fuzzy metrics that would be useful in dealing with discrepancies among different scanners detecting vulnerabilities in the system. The vulnerability with a higher TP value can be used to determine the certainty of the vulnerability being present and a high TN value indicates a high probability that the vulnerability is absent.

The fuzzy classifiers are defined as,

$$VI(j)_T = \sum_{i=1}^{WS_T} I_{ij} \quad \textbf{(1)}$$

$$VI_F(j)_T = \sum_{i=1}^{WS_T} F_{ij} \quad \textbf{(2)}$$

$$V(j)_{TP} = \frac{\sum_{k=1}^{WS_T}\sum_{i=1}^{I_{kj}} d(v_{ij})}{VI(j)_T} = \frac{\sum_{i=1}^{VI(j)_T} d(v_{ij})}{VI(j)_T} \quad \textbf{(3)}$$

$$V(j)_{FP} = \frac{\sum_{k=1}^{WS_T}\sum_{i=1}^{F_{kj}} d(v_{ij})}{VI_F(j)_T} = \frac{\sum_{i=1}^{VI_F(j)_T} d(v_{ij})}{VI_F(j)_T} \quad \textbf{(4)}$$

$$V(j)_{TN} = 1 - V(j)_{FP} \quad \textbf{(5)}$$

$$V(j)_{FN} = 1 - V(j)_{TP} \quad \textbf{(6)}$$

Where,
$D(v_{ij})$={1 if instance of vulnerability j is detected at location *i* or
0 if instance of vulnerability *j* is not detected at location *i* }
I_{ij} is the number of instances of vulnerability *j* present at location *i* during calibration
F_{ij} is the number of instances of vulnerability *j* falsely detected at location *i* during calibration
$VI(j)_T$ Is the total number of instances of vulnerability j used for calibration
$VI_F(j)_T$ is the total number of instances of vulnerability *j* falsely detected during calibration

WS_T is the total number of test websites used for calibration

$V(j)_{TP}, V(j)_{TN}, V(j)_{FP} \& V(j)_{FN}$ are the vulnerability specific truth values which refer to the vulnerability specific true positive, true negative, false positive and false negative, respectively.

C. Scanner Specific Truth-Values

Different scanner's output data differ in content, format and organization [1, ICIMP 2009]. The data generated by scanners depends, to an extent, on the algorithm being used in the specific scanner. Some scanners with access to large databases are equipped to detect more classes of vulnerabilities. Others comprising lightweight computational modules provide basic diagnoses while several lie somewhere in between. For example, IBM AppScan and HP WebInspect are scanners with access to large databases while NStalker is associated with a relatively smaller database. It is hence necessary to first analyze and understand the scanning process as well as the capability of the scanner in order to derive the required metrics.

Let the scanner specific truth-values be represented by { S_{TP}, S_{TN}, S_{FP}, S_{FN} }. These are also known as {True Positive, True Negative, False Positive and False Negative}, respectively. These values form an important measure of the vulnerability detection capability of the scanner. Derived scanner metrics require calibration in order to grade the scanner prior to its usage for adequate quality assurance based on performance. Calibration of the detection capabilities of the scanner is performed with a sample of customized websites for positive or negative vulnerabilities thus reflecting the performance of a scanner with expected results providing a valid basis for a suitable quality check.

In the pre-calibration phase, the system would be unlikely to produce results with the levels of reliability expected by the user. The scanner metrics are defined with respect to a scanner's prediction capabilities. The prediction capabilities of the scanner are then calibrated against the expected prediction performance. Vulnerabilities are also calibrated in this phase and their fuzzy metrics are defined in section V (C.). It must be noted that the classification of vulnerabilities in the calibration phase influences the scoring by CVSS [7]. Once the *calibration phase* for the scanner is completed, the reports from the scanner can be processed for data in a more reliable manner. The fuzzy metrics for scanners are defined as follows:

$$U_T = \sum_{j=1}^{EV_T} VI(j)_T \quad \textbf{(7)}$$

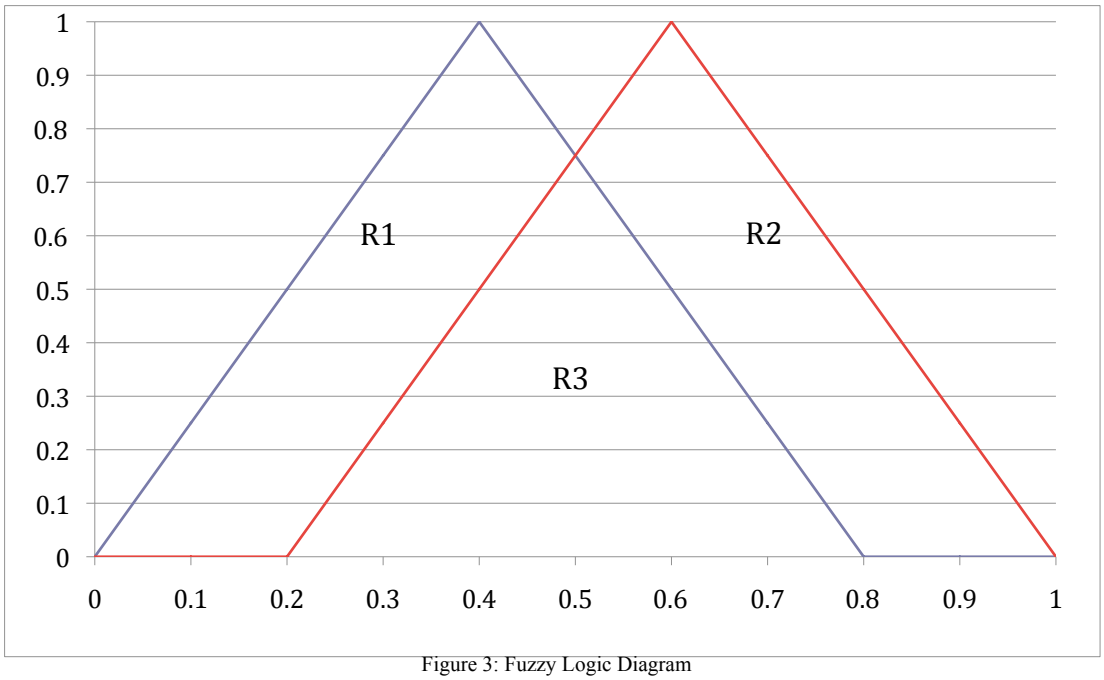

Figure 3: Fuzzy Logic Diagram

$$U_F = \sum_{j=1}^{EV_T} \sum_{i=1}^{WS_T} U(F_{ij}) \quad \textbf{(8)}$$

$$S_{TP} = \frac{\sum_{j=1}^{U_T} V(j)_{TP}}{U_T} \quad (9)$$

$$S_{TN} = \frac{\sum_{j=1}^{U_F} V(j)_{TN}}{U_F} \quad (10)$$

$$S_{FP} = 1 - S_{TN} \quad (11)$$

$$S_{FN} = 1 - S_{TP} \quad (12)$$

Where,

U_T is the total number of unique vulnerabilities incorporated for calibration.

U_F is the total number of unique vulnerabilities falsely detected by scanners.

$U(F_{ij})$={1, if the instance is unique for that vulnerability || 0, otherwise}

EV_T is the total number of vulnerability instances evaluated

$V(j)_{TP}$ and $V(j)_{TN}$ are vulnerability specific truth values defined in section V.B

$VI(j)_T$ is the total number of instances of vulnerability j used for calibration

S_{TP}, S_{TN}, S_{FP}&S_{FN} are the scanner specific truth values.

Figure 3 shows the graphical representation of the fuzzy metrics for the scanner. A similar diagram can also be used for vulnerability fuzzy metrics. The region R1 represents the true positive S_{TP}, the region R_2 represents the true negative S_{TN} and the region R_3 is the combined space of false positive and false negative S_{FP} and S_{FN}.

VI. GRADING SYSTEMS

This section describes the design details of the two grading systems: the scanner grading system and the vulnerability grading system.

A. Scanner Grading System

The scanner grading system is used to grade the capability of a web application scanner. The scanner grading system makes use of a scanner database as well as the vulnerability databases [17]. The scanner database comprises a table list that is maintained for every graded scanner (see Figure 4). It contains information on the scanner specific truth values as well as the vulnerability specific truth values. The Scanner Grade, S_{GRADE}, may be computed for all web-based vulnerabilities listed in the vulnerability database.

The overall *sensitivity* and *specificity* of the scanners can be computed by using the equations, Eqn. 13 and Eqn. 14. Sensitivity is the percentage of correctly detected activities out of all true instances of a particular class, averaged over

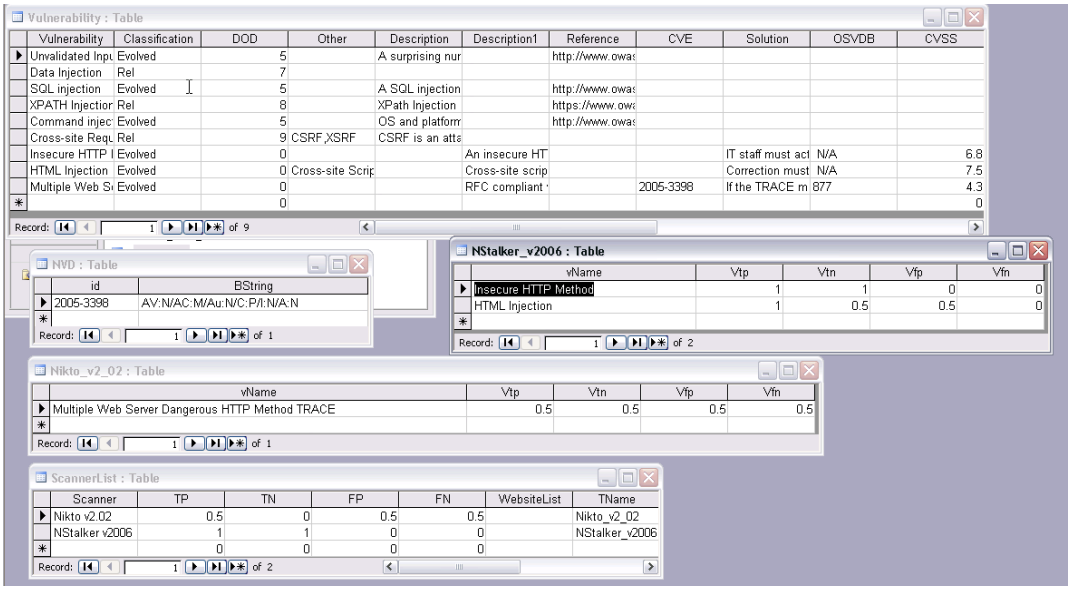

Figure 4: Table List in Scanner Grading System

all activities. Specificity measures the proportion of correctly identified negative occurrences to all true negative occurrences. If a scanner is more sensitive, it has a greater chance of discovering the vulnerability. Similarly, if a scanner is more specific, it has a greater chance of discovering the absence of the vulnerability.

$$S_{Specificity} = \frac{S_{TP}}{S_{TP} + S_{FN}} \quad \textbf{(13)}$$

$$S_{Specificity} = \frac{S_{TN}}{S_{TN} + S_{FP}} \quad \textbf{(14)}$$

Where,
S_{TP}, S_{TN}, S_{FP}&S_{FN} are the scanner specific truth values defined in section 5.3
$S_{ensitivity}$ is the sensitivity measure of the scanner
$S_{Specificity}$ is the specificity measure of the scanner
However, a greater sensitivity could also mean greater probability of false positives for the scanner. Similarly, a higher specificity could mean there are a greater number of false negatives for the scanner. For a given scanner, the trade-off between sensitivity and specificity depends on the vulnerability and the web application being scanned.

B.Vulnerability Grading System:

As mentioned previously, the representation for the vulnerability specific truth values is also similar to Figure 3. The *vulnerability specific sensitivity* and *specificity* for a scanner are defined by Eqns. 15 and 16, respectively. We also define the *likelihood ratio* for both true positive and true negative results with the Eqns. 17 and 18, respectively.

$$V_{Sensitivity} = \frac{V_{TP}}{V_{TP} + V_{FN}} \quad \textbf{(15)}$$

$$V_{Specifi\ city} = \frac{V_{TN}}{V_{TN} + V_{FP}} \quad \textbf{(16)}$$

$$V_{LR+} = \frac{V_{TP}(V_{TN} + V_{FP})}{V_{FP}(V_{TP} + V_{FN})} \quad \textbf{(17)}$$

$$V_{LR-} = \frac{V_{FN}(V_{TN} + V_{FP})}{V_{TN}(V_{TP} + V_{FN})} \quad \textbf{(18)}$$

Where,
$V_{Sensitivity}$is the vulnerability specific sensitivity measure for the scanner
$V_{Specificity}$is the vulnerability specific specificity measure for the scanner
$V_{LR+}$is the likelihood ratio for positive detection
V_{LR-} is the likelihood ratio for negative detection

V_{LR+} in Eqn. 17 gives the likelihood ratio of the vulnerability to be present given the vulnerability specific truth values for the specified scanner. V_{LR-} in Eqn. 18 gives the likelihood ratio of the vulnerability to be absent given the vulnerability specific truth-values for the specified scanner. Combined with the scanner metrics, these could be used as a basis in predicting the levels of vulnerability present. However, this will hold absolutely true only for vulnerabilities that fall under the evolved-vulnerability category.

The vulnerability specific sensitivity and the specificity metrics can also be used to study the scanner's performance and behavioral characteristics with certain classes of vulnerabilities. If the scanner is more sensitive towards a specific vulnerability, it will exhibit better detection of the presence of that particular vulnerability and if it is more specific, it will be more able to detect the absence of the particular vulnerability.

A *positive vulnerability* means that the vulnerability is present in the website location at that instant and there is evidence to support it. A *negative vulnerability* means that the vulnerability is not present in the website and can be proved to a satisfactory level. Assertion 3 implies that some vulnerabilities may be more difficult to find and may generate false negatives. Similarly, some vulnerabilities may be more complex and can lead to the generation of false positives. Hence, it is important to grade each vulnerability to an adequate level.

Assertion 1 states the need to grade the various vulnerabilities with a vulnerability grading system. The system could create a list of known web-based vulnerabilities from the online vulnerability databases [15], classified into evolved, dormant, relatively-new or new categories as defined in Assertion 2.

Such a grading would also provide a better understanding of the vulnerability. The grading is a constantly changing one as the scanner algorithms may change over time with upgrades and there are also instances that the vulnerability definitions themselves may change [23].

The *difficulty of detection* of a vulnerability *j* is given by

$$D(j) = 1 - \frac{\sum_{s=1}^{S_n} (V(j)_{TN} + V(j)_{TP}) / 2}{S_n} \quad \textbf{(19)}$$

Where,
D(j) is the difficulty of detection of vulnerability *j*
S_n is the number of scanners used
$V(j)_{TN}$ is the vulnerability specific true negative value for vulnerability *j* for a specific scanner
$V(j)_{TP}$ is the vulnerability specific true positive value for vulnerability *j* for a specific scanner
The higher the value of D(j), the more difficult it is to detect and the lower difficulty implies the easier detection by the scanner.

TABLE 1: DETECTION EXAMPLE 1

Vulnerability	Scanner1	Scanner2	Scanner3
Vul1	YES	YES	YES
Vul2	YES	NO	YES
Vul3	NO	NO	NO
Vul4	YES	NO	YES
! Vul1	NO	NO	NO
! Vul2	NO	NO	YES
! Vul3	NO	YES	NO
! Vul4	YES	YES	NO

In the table 1, "VulX" refers to the positive vulnerability and "! VulX" refers to the negative vulnerability. Table 1 has been formed with the assumption that all the scanners used in the calibration have been designed with their respective algorithms to detect the stated vulnerabilities.
Applying Eqn. 19 to table 1, we can get
D(Vul1)=0; D(Vul2)=0.333; D(Vul3)=0.666; D(Vul4)=0.5;
From the above table we can conclude that the D(Vul3)>D(Vul4)>D(Vul2)>D(Vul1)

VII. COMPUTATION OF SCANNER REPORT CONFIDENCE

While it is important for the end-user to be able to infer from various diagnostic reports, it is also important to be able to gauge the confidence of the information within the report. Confidence levels are required to ascertain if the report can be trusted and the extent of this trust. There needs to be a confidence level associated with every vulnerability detected by the various scanners. The grading systems are used in the calculation of the confidence of the report.

It must be noted that not all scanners agree on the reports generated. Some scanner algorithms may be better suited to tackle some classes of vulnerability and they are effective against these but the same algorithm may be the reason for their weaker performance against other classes of vulnerabilities. Some scanners have access to a very comprehensive database while others suffer from inadequate ones. Hence, using the S_{GRADE} (defined in Assertion 3) we can specify the performance of the scanner with respect to a particular vulnerability. When the results of all scanners agree on that vulnerability, the confidence on the report will be higher than the confidence due to an individual scanner. This factor is also moderated when scanners have a conflict in results. The moderation, however, depends on the metric values assigned to the scanners by the scanner grading system.

The methodology for calculating the 1st and 2nd degree confidence is detailed as follows :

To find 1st Degree Confidence.

The 1st degree confidence report gives a report based on the various scanner reports on a website location. The various scanner and vulnerability specific truth-values are used along with the vulnerability reports to give a more useful inference to the user.

There are two types of indices that the 1st Degree Confidence report can have. They are the *positive index* and the *negative index*. The report first tries to calculate the positive index. This represents the possibility of the vulnerability being present. It shows how confident the user can be about the vulnerability being present in a scale ranging from (>0%) to 100%. Only if the condition given in step 7 of positive index is satisfied, the negative index is calculated. The negative index represents the possibility of vulnerability not being present. The computed output will contain only one of the two indices. If any scanner is known to lack the detection capability with regard to the vulnerability, then the scanner must be excluded from the computation.

Positive Index Computation:

1. Create the set A that contains the respective Vulnerability specific V_{TP} or V_{TN} values depending on whether the scanner detects the vulnerability or not.
2. Let a_0={largest V_{TP} value in A}. If there are no V_{TP} values calculate negative index.
3. $A \leftarrow - A - a_0$, Where – refers to set subtraction.
4. Rearrange the remaining elements in descending order.
5. $A=a_0 \pm a_1(1-a_0) \pm (a_2(1-(a_0 \pm a_1(1-a_0))) \pm \ldots\ldots\ldots\ldots (a_n(1-(a_{n-1}(\ldots(1-a_2(1-(a_0 \pm a_1(1-a_0))))\ldots))))$ where **(20)**
   - a gives the 1st degree confidence report
   - $a_i \in A \;\&\; (0 \le a_i < 1)$

6. If a_i is a V_{TP} value, it is added otherwise, it is subtracted.
7. If $a \leq 0$ calculate negative index.
8. The resultant value will give the confidence level for the vulnerability being present from the scanner reports obtained.

Figure 5. Relationship example

Negative Index Computation:

1. Create the set A that contains the respective Vulnerability specific V_{TP} or V_{TN} values depending on whether the scanner detects the vulnerability or not.
2. A0={largest V_{TN} value in A}
3. $A \leftarrow - A - a_0$, Where – refers to set subtraction.
4. Rearrange the remaining elements in descending order.
5. $A=a_0 \pm a_1(1-a_0) \pm (a_2(1-(a_0 \pm a_1(1-a_0))) \pm \ldots\ldots\ldots\ldots (a_n(1-(a_{n-1}(\ldots(1-a_2(1-(a_0 \pm a_1(1-a_0))))\ldots))))$ where (21)
   - a gives the 1st degree confidence report
   - $a_i \in A \;\&\; (0 \leq a_i < 1)$
6. If a_i is a V_{TN} value, it is added otherwise, it is subtracted.
7. The resultant value will give the confidence level for the vulnerability being absent from the scanner reports obtained.

The value of a is the positive or negative index that is associated with the report. The general idea behind the derivation of a lies in the fact that the positive and negative indices can never be ~~a~~ 100%. While the rule in itself does not stop them from taking the value of 0%, it is that this value could appear. This is because being 0% requires both positive and negative indices to be 0 i.e. sufficient scanners to nullify the true detection and sufficient scanners to nullify the false detection. Hence, the value of a is restricted to be always less than 1.

To find 2nd Degree Confidence.

A 1st degree confidence report covers the direct existence of any vulnerability. However, a 2nd degree report is necessary to cover Assertion 5. The 2nd Degree Confidence covers the fact that one vulnerability can have a relationship to or influence the existence of another.

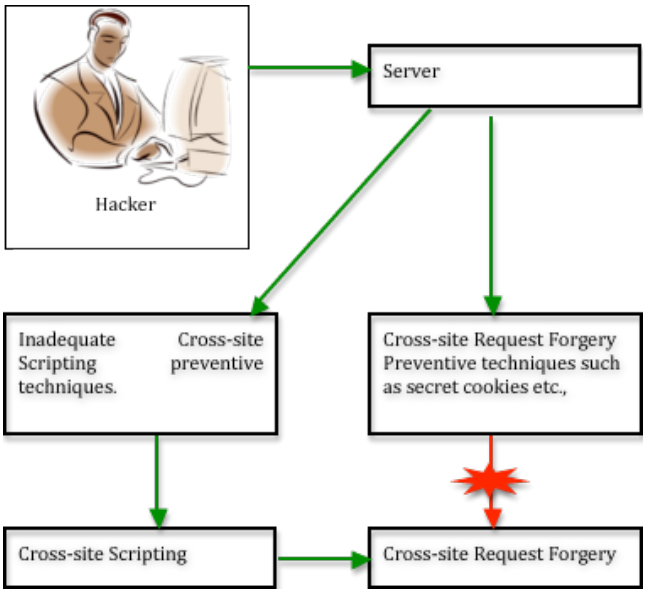

Figure 6: XSS-CSRF Example

The 2nd Degree Confidence report is hence a result of combining the confidence reports with the Assertion 5. The relationship between two vulnerabilities is given by a value ranging between 0 and 1. Every vulnerability has a relationship of 1 with itself. The relationship with other vulnerabilities is generally less than 1. However, there could be exceptions as exemplified by Figure 5. If the vulnerability "Use of a Resource after Expiration or Release" is present, then the vulnerability "Improper Validation of Certificate Expiration" has a lower likelihood of presence but if the reverse situation occurs, the presence of the vulnerability "Improper Validation of Certificate Expiration" would mean the same (if not greater) likelihood of the presence of "Use of a Resource after Expiration or Release". i.e. "Improper Validation of Certificate Expiration" has a relationship of 1 towards "Use of a Resource after Expiration or Release" but "Use of a Resource after Expiration or Release" has a relationship of

<1 towards "Improper Validation of Certificate Expiration". Quality of the final result is enhanced by the use of these relations since they take more factors into account than a typical scanner-based risk analysis that are available currently [6].

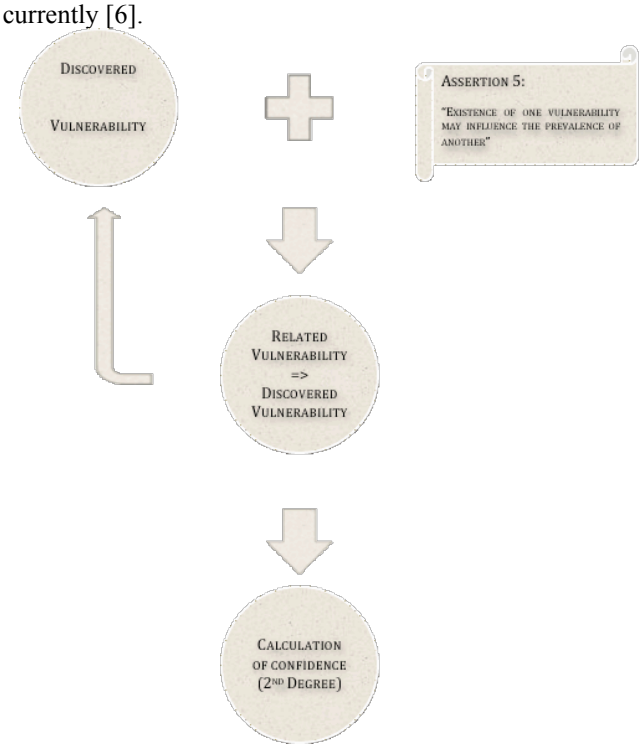

Figure 7: Calculation of 2nd degree report

From the Figure 5, the relation of "Failure to Follow Chain of Trust in Certificate Validation" to "Failure to Validate Certificate Expiration" can be defined as that of a subset-superset where the former is the superset and the latter is the subset. Such a relationship would also contain the fact that the absence of the former does not mean the absence of the latter. Another example is that of the Cross-site scripting and Cross-site request forgery.

The Cross-site request forgery (CSRF or XSRF) and Cross-site scripting (XSS) example is one that shows how influential the relationships are in a system. From Figure 6, it can be seen that even while CSRF is prevented for the hacker, if XSS is not prevented, the CSRF cannot be considered as an eliminated threat. It should also be noted that the reverse is not true.

The Figure 7 shows the calculation of the 2nd degree confidence report from the 1st degree report.

Let there be a list of related vulnerabilities for the vulnerability v. This list includes both the ones that are directly related and those that are indirectly related till a satisfactory set of related vulnerabilities is formed.

Let the a(v) be the 1st degree confidence of vulnerability v. If a(v) is a positive index value, then $a_0(v)=a(v)$. If a(v) is a negative index value, then $a_0(v)=0$.

The above step is the pre-operation step that needs to be done to ensure that the fact that the steps taken against one vulnerability do not reflect on the decrease in the 2nd degree confidence if the Assertion 5 relates to it through another vulnerability known to be present. The extent of confidence on the vulnerability's presence, however, depends on the strength of the relationship value and the confidence level on the vulnerability relating to it. A chain of related vulnerabilities could be formed by relating one vulnerability to another that is related to another and so on.

The Figure 8 shows the relationship between various vulnerabilities obtained from a recent analysis.

$$\begin{aligned} a_1(v) &= a_0(v) + \\ &(1 - a_0(v))(R_v(v_1) * a_0(v_1)) + \\ &(1 - ((1 - a_0(v))(R_v(v_1) * a_0(v_1)))(R_v(v_2) * a_0(v_2))) + \\ &(1 - (1 - ((1 - a_0(v))(R_v(v_1) * a_0(v_1)))(R_v(v_2) * a_0(v_2))))(R_v(v_3) * a_0(v_3)) + ... \end{aligned} \quad \textbf{(22)}$$

Using the above equation, the $a_1(v)$ is calculated for the entire set of related vulnerabilities {R}. In case of a closed relation eg. v related to v1, v1 related to v2 and v2 related to v, then v is not considered related to v2 during calibration of v.

As the relationship keeps expanding eg.v related to v1, v1 related to v2 etc., the same procedure is repeated for $a_2(v)$ replacing all $a_0(v)$ with $a_1(v)$, then $a_3(v)$ and so on. It is repeated till $a_n(v)$ is calculated. This gives the 2nd degree confidence level.

The confidence levels have been found to be a useful criteria for evaluation of any given website. A remediation database is being designed to give a suitable remediation based on the confidence level. Hence a more accurate remediation can be provided to the user.

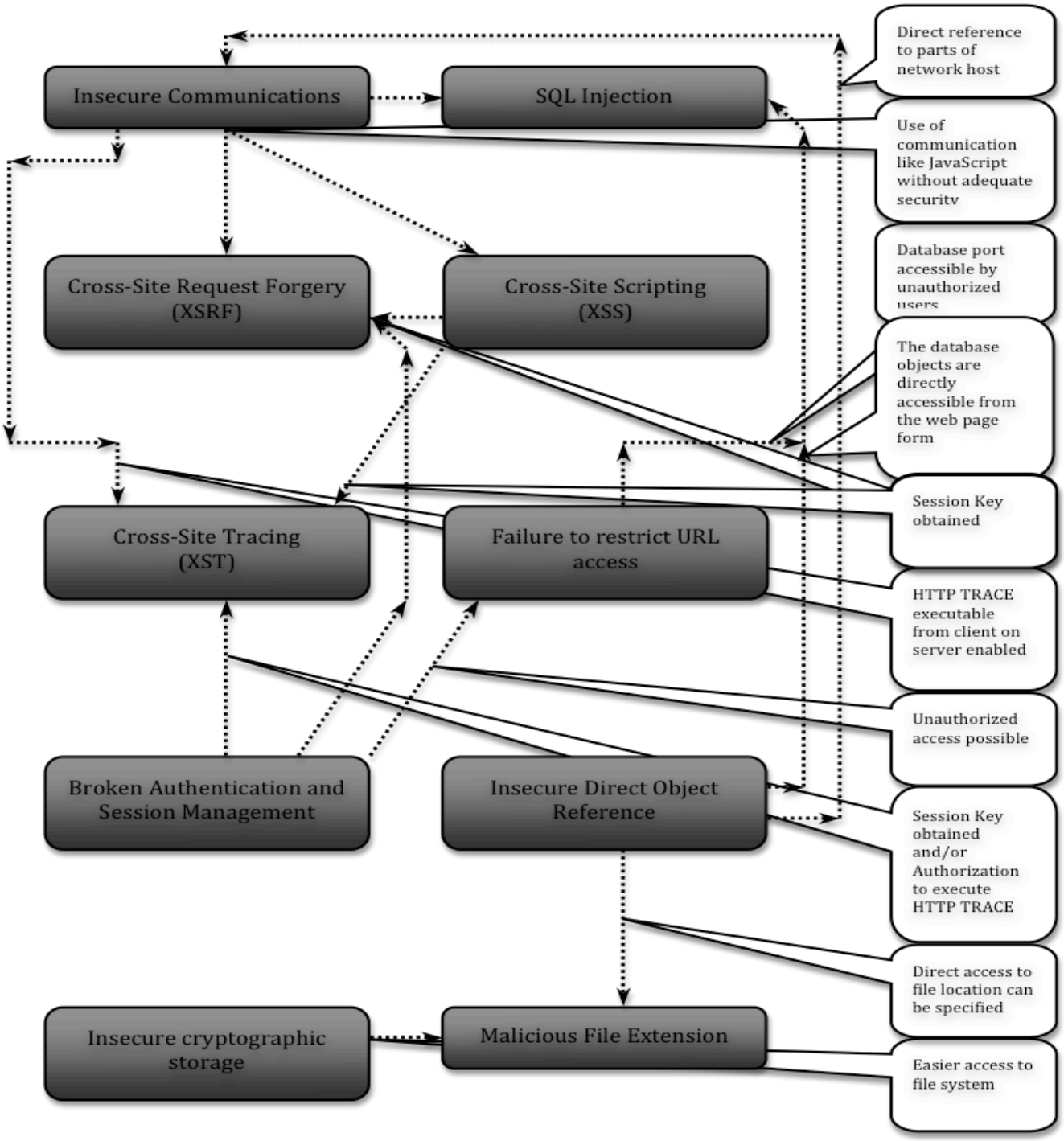

Figure 8: Analysis of relationship between vulnerabilities

VIII. ILLUSTRATIVE EXAMPLE

Let there be 3 scanners s_1, s_2, s_3. Hence S_n=3.
Let there be 5 vulnerabilities detected and stored in the database v_1 v_2 v_3 v_4 v_5.

TABLE 2: S-V TABLE

	s_1		s_2		s_3	
	Instance1	Instance 2	Instance1	Instance 2	Instance1	Instance 2
v_1	Y	Y	Y	Y	Y	Y
v_2	Y	Y	Y	Y	Y	N
v_3	Y	Y	N	N	Y	Y
v_4	Y	Y	Y	N	N	N
v_5	Y	N	N	N	N	Y

TABLE 3: S-!V TABLE

	s_1		s_2		s_3	
	Instance1	Instance 2	Instance1	Instance 2	Instance1	Instance 2
! v_1	N	N	N	N	N	N
! v_2	N	N	N	N	N	Y
! v_3	N	N	Y	Y	N	N
! v_4	N	N	N	Y	Y	Y
! v_5	N	Y	Y	Y	Y	Y

Table 2 shows the detection for instances where the vulnerability is known to be present. Table 3 shows the detection where the vulnerability is known to be absent.

TABLE 4: VULNERABILITY SPECIFIC TRUTH-VALUES

	v_1	v_2	v_3	v_4	v_5
s_1	{.9 .9 .1 .1}	{.92 .8 .08 .2}	{.8 .7 .2 .3}	{.8 .7 .2 .3}	{.7 .6 .3 .4}
s_2	{.8 .7 .2 .3}	{.7 .6 .3 .4}	{.7 .6 .3 .4}	{.6 .6 .4 .4}	{.5 .5 .5 .5}
s_3	{.7 .6 .3 .4}	{.8 .71 .2 .3}	{.6 .6 .4 .4}	{.9 .8 .1 .2}	{.7 .6 .3 .4}

The Vulnerability specific truth-values in table 4 are represented as $\{V_{TP}, V_{TN}, V_{FN}, V_{FP}\}$

$D(v_1)=1-(6/6)=0$
$D(v_2)=1-(5/6)=1/6=0.1667$
$D(v_3)=1-(2/3)=1/3=0.3333$
$D(v_4)=1-(1/2)=1/2=0.5$
$D(v_5)=1-(1/4)=3/4=0.75$

Hence $D(v_5)>D(v_4)>D(v_3)>D(v_2)>D(v_1)$

1st degree confidence.

In this example let us consider the 2nd instance of v_2 from the table 2.

The results from the 3 scanners are {Y Y N}
Hence A={.92 .71 .7} and a_0=.92
Hence A={.71 .7}
a=.92-(.08(.71))+(1-.(92-(.08(.71))))(.7)=0.95896
In other words, we can be 95.896% certain that the result is true. This could mean there is high need for appropriate remediation.

2nd degree confidence.

TABLE 5: R-V TABLE

	v_1	v_2	v_3	v_4	v_5
R_{v1}	-	-	-	-	-
R_{v2}	.08	-	-	-	-
R_{v3}	-	-	-	.05	.04
R_{v4}	-	-	-	-	.09
R_{v5}	-	-	.1	-	-

The Table 5 shows the relationship between the various vulnerabilities and the suitable relationship values.

Let us consider the 2nd instance in the Table 2 and calculate the confidence for vulnerability v_2. Since only v_1 is related to v_2 (There are no indirect relations as v_1 is not related to any other vulnerability).

We know that for 2nd instance in Table 2,
$a_0(v_1)$ =.987{Y YY , all scanners have detected the vulnerability as positive}
$a_0(v)$ =.95896
$a_1(v)$ =.95896+.040506=.9994

Hence, the 2nd degree confidence report would suggest a possibility of 99.94% for the vulnerability's occurrence. Since $a_1(v)>a_0(v)$, it implies that other vulnerabilities are also present and must be rectified to rectify this vulnerability.

Let us consider another example using the 1st instance from table 2.

For the vulnerability v_5, A={Y N N}

1st Degree Confidence,
$a_0(v_5)$ =.7-(.3(.6))-((1-(.7-(.3(.6))))(.5)) =0.28

It can be observed that when there is only one scanner supporting the vulnerability, the confidence of report in the presence of the vulnerability also drops to a great extent. In the case shown above, the confidence report still remains in the positive index.

2nd Degree Confidence,

The vulnerability v_5 is related to v_3, which in turn is related to v_4 and v_5, and v_4 is related to v_5. The 1st degree confidences for these vulnerabilities can be given for the 1st instance of table 2 as

$a_0(v_3)$ =.8+(.2(.6))-(1-(.8+(.2(.6))))(.6) =0.872
$a_0(v_4)$ =.8-(.2(.8))+(1-(.8-(.2(.8))))(.6) =0.856

$v=v_5$

$a_0(v)$ =0.28
$a_1(v)$ =$a_0(v)$+(1-$a_0(v)$)(0.1($a_0(v_3)$))=0.28+(0.72)(0.1(0.872))=0.343
$a_2(v)$ =$a_1(v)$+(1-$a_1(v)$)(0.1($a_1(v_3)$))

$a_1(v_3)$ can be given by,
$a_1(v_3)$ =$a_0(v_3)$+(1-$a_0(v_3)$)(.05($a_0(v_4)$)+(1-(1-$a_0(v_3)$)(.05($a_0(v_4)$)))(0.04($a_0(v_5)$))
but, since v=v5, the underlined portion of the equation above cannot be considered
hence, a1(v3) becomes
=0.872+(.128)(.05(0.856))
=0.877

therefore,
$a_2(v)$ =0.343+0.657(0.1(0.877)) = 0.401

Hence the 2nd Degree Confidence report shows 40.1% confidence in the presence of vulnerability v5 compared to the 28% confidence showed by the 1st degree confidence report.

IX. Conclusion

This research has enabled the improved risk analysis of web-based vulnerabilities. While several scanners are available to detect the vulnerabilities, their varying algorithms and proprietary nature makes it difficult to ascertain if the vulnerabilities found by them is true or false. The methodology used in this paper is a practical approach designed to work in spite of the proprietary nature of the algorithms while still being able to grade the various scanners. The variable nature of a vulnerability is also accounted for and the proposed methodology uses fuzzy-based classification and estimation metrics solve this problem. Another problem is the lack of detection of vulnerabilities whose presence is also influenced by other vulnerabilities. The proposed methodology is an effective one to tackle such problems as well. Five assertions have been defined to help establish the theoretical aspects of the approach. By using relationship between vulnerabilities given by assertion 5, the 2nd degree confidence report can be used to tackle such a problem. The confidence reports have been able to provide the user with valuable information and this has been tested with a successful implementation of the system. The confidence reports also provide a greater reliability of results than that of individual scanner reports. The open issues faced in this research include the need to grade every scanner for every vulnerability using test sites, which can be a very tedious process. The methodology of finding the relationship between vulnerabilities is still in progress. The development of an inference engine to compute the diagnosis is also in progress. The remediation database to accurately provide the remediation based on confidence level is being expanded to a sizable level of vulnerabilities.

References

[1] D. Subramanian, H.T. Le, P.K.K. Loh, "Fuzzy Heuristic Design For Diagnosis Of Web-Based Vulnerabilities", *The Fourth International Conference on Internet Monitoring and Protection* (ICIMP), May 2009, Venice/Mestre, Italy.

[2] H. T. Le and P. K. K. Loh, "Unified Approach to Vulnerability Analysis of Web Applications," in *International Electronic Conference on Computer Science. AIP Conference Proceedings*, Volume 1060, pp. 155-159 (2008)

[3] H.-T. Le and P. K. K. Loh, "Realizing Web Application Vulnerability Analysis via AVDL," in *10th International Conference on Enterprise Information Systems (ICEIS 2008)*, Barcelona, Spain, 2008, pp. 259-265.

[4] H. T. Le and P. K. K. Loh, "Evaluating AVDL Descriptions for Web Application Vulnerability Analysis," in *IEEE International Conference on Intelligence and Security Informatics 2008 (IEEE ISI 2008)*, Taipei, Taiwan, 2008, pp. 279-281.

[5] Jonathan Gomez and Dipankar Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection", *Proceedings of the 3rd Annual IEEE Information Assurance Workshop*, New Orleans, Louisiana: June, 2002.

[6] Larry Suto, "Analyzing the Effectiveness and Coverage of Web Application Security Scanners", White Paper, October 2007, Publisher: Strategic Data Command, http://www.stratdat.com/webscan.pdf

[7] P. Mell, K. Scarfone, and S. Romanosky, "The Common Vulnerability Scoring System (CVSS) and its applicability to federal agency systems," NIST Interagency Report, NIST-IR7435, pg. 1-20, August 2007, http://csrc.nist.gov/publications/nistir/ir7435/NISTIR-7435.pdf

[8] Quals, "Vulnerability Management for Dummies", *Copyright © 2008 by John Wiley & Sons Ltd*, Chichester, West Sussex, England.

[9] Ramaswamy Chandramouli, Tim Grace, Rick Kuhn and Susan Landau, "Emerging Standards: Common Vulnerability Scoring System", IEEE Security & Privacy, vol. 4, no. 6, pp. 85-89, Nov/Dec, 2006.

[10] David Minnen, Tracy Westeyn, Thad Starner, Jamie A. Ward and Paul Lukowicz, "Performance Metrics and Evaluation Issues for Continuous Activity Recognition", Performance Metrics for Intelligent Systems (PerMis'06), Gaithersburg, Maryland, United States of America, August 21-23, 2006, www.cc.gatech.edu/~dminn/papers/minnen-permis2006.pdf

[11] Ambareen Siraj, Susan M. Bridges, Rayford B. Vaughn, "Fuzzy Cognitive Maps for Decision Support in an Intelligent Intrusion Detection System", *Proceedings of the Joint 9th International Fuzzy Systems Association World Congress and the 20th North American Fuzzy Information Processing Society International Conference on Fuzziness and Soft Computing in the New Millennium*, vol. 4, pg. 2165-2170, July 2001, Vancouver, Canada,

[12] Jeffrey R. Jones, "Estimating Software Vulnerabilities", *IEEE Security & Privacy*, Volume 5, Issue 4, pg. 28 – 32, July-Aug 2007.

[13] Omar H. Alhazmi, Yashwant K. Malaiya, "Quantitative Vulnerability Assessment of Systems Software", *Proceedings of the Annual Reliability and Maintainability Symposium*, pg. 615-620, Jan. 2005.

[14] P. Wang, K.-M. Chao, C.-C. Lo, C.-L. Huang, and M. Younas, "A fuzzy outranking approach in risk analysis of web service security ", *Cluster Computing*, vol. 10, pp. 47-55, 2007.

[15] S. T. Halkidis, A. Chatzigeorgiou, and G. Stephanides, "Quantitative Evaluation of Systems with Security Patterns Using a Fuzzy Approach", *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*: Springer Berlin / Heidelberg, 2006, pp. 554-564.

[16] Kenneth L. Ingham, Anil Somayaji, John Burge and Stephanie Forrest, "Learning DFA representations of HTTP for protecting web applications", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Volume 51 , Issue 5, pg. 1239-1255 , April 2007

URLs

[17] CERT/CC Statistics 1988 – 2006, http://www.cert.org/stats/

[18] National Vulnerability Database (NVD) Statistics, http://nvd.nist.gov/statistics.cfm

[19] DHS National Security Division, NIST, "Web Application Vulnerability Scanners", https://samate.nist.gov/index.php/Web_Application_Vulnerability_Scanners

[20] Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/

[21] Jeremiah Grossman, "WhiteHat Website Security Statistics Report", October 2007, Publisher: WhiteHat Security (United States of America). https://whitehatsec.market2lead.com/go/whitehatsec/WPStatsreport_100107

[22] Jeremiah Grossman, "WhiteHat Website Security Statistics Report", August 2008, Publisher: WhiteHat Security (United States of America). https://whitehatsec.market2lead.com/go/whitehatsec/WPstats0808

[23] Jeremiah Grossman, "WhiteHat presentation on XSRF", Publisher: WhiteHat Security (United States of America) https://whitehatsec.webex.com/whitehatsec/nbrshared.php?action=playback&recordID=21578512&recordKey=2E8BF7FFE53556F277FD706294A7E3ED86F81580F46B5A0DDC7345881C9B224C

[24] N-Stalker®, "Overview of N-Stalker Reports", http://nstalker.com/products/development/report-details, Retrieved on 20th July 2009

[25] Accunetix web application Security, "In depth checking for SQL Injection, Cross Site Scripting (XSS) and Other Vulnerabilities", http://www.acunetix.com/vulnerability-scanner/sql-injection-ft.htm, Retrieved on 20th July 2009

[26] Wikipedia, XSS, http://en.wikipedia.org/wiki/Cross-site_scripting, Retrievedon 20th July 2009.