SOFTWARE SECURITY RISK ANALYSIS USING FUZZY EXPERT SYSTEM

SODIYA A. S.[1]
LONGE H. O. D.[2]
FASAN O. M.[3]

UNIVERSITY OF AGRICULTURE
DCC - Department of Computer Science
P. M. B. 2240
ABEOKUTA, NIGERIA[1], 3
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF LAGOS
AKOKA, LAGOS[2]
[1]sinaronke@yahoo.co.uk
[2]hodlonge@yahoo.co.uk
[3]sola_fasan@mail.com

Abstract. Today, there is wide concern on the security of software systems because many organizations depend largely on them for their day-to-day operations. Since we have not seen a software system that is completely secure, there is need to analyze and determine the security risk of emerging software systems. This work presents a technique for analyzing software security using fuzzy expert system. The inputs to the system are suitable fuzzy sets representing linguistic values for software security goals of confidentiality, integrity and availability. The expert rules were constructed using the Mamdani fuzzy reasoning in order to adequately analyse the inputs. The defuzzification technique was done using Centroid technique. The implementation of the design is done using MATLAB fuzzy logic tool because of its ability to implement fuzzy based systems. Using newly develop software products from three software development organizations as test cases, the results show a system that can be used to effectively analyze software security risk.

Keywords: Software systems, Software security, Risk analysis, Fuzzy logic

(Received January 18, 2008 / Accepted July 16, 2008)

1 INTRODUCTION

Nowadays, security problem involving computer-based systems are getting more frequent and this calls for serious attention. The number and variety of attacks by person and malicious software from outside organization, particularly via the Internet are rapidly increasing. The amount and consequences of inside attacks also remain a major concern.

However, it is widely accepted that designing secure software is a difficult problem. Attackers break routinely into systems and in response, some software vendors started providing security as a necessary feature for their products and network systems. As a result of years of research in computer security, many powerful techniques have been developed to solve a wide array of security problems. Yet, some security features in some software are either unnecessary or the software does not meets its security needs. Infact, producing secure software products requires a lot of security engineering [9].

Overtime, a major phase that has been incorporated into the software development in order to improve software security is the security testing stage, which basically uses a risk-based security test. Since perfect security appears to be an unattainable goal with the present technologies, producing secure software becomes a question of risk management. Security risk must be identified, ranked, mitigated and then must be managed throughout the software product lifecycle. Once the security risks are identified, adjustments are then made to the software before the final deployment.

Thus, this work proposes a fuzzy logic-based technique for determination of level of security risk associated with software systems. Fuzzy logic, as introduced in [10], is one of the major tools used for security analysis. The major goals of secure software which are used as the inputs to the system are: the preservation of confidentiality (preventing unauthorized disclosure of information), preservation of integrity (preventing unauthorized alteration of information) and preservation of availability (preventing unauthorized destruction or denial of access or service to an authentic user).

The rest of this paper is organized as follows. Section 2 presents related works. In section 3, the methodology of design is described. Section 4 explains the implementation procedure and section 5 contains the evaluation of the system. The future work and conclusion are presented in section 6.

2 RELATED WORKS

Despite all the efforts towards producing high quality and secure software systems, attackers still frequently break into these systems. In response, software vendors started incorporating security as a necessary feature for their products. Software security is critical to information assurance and design-level vulnerabilities are responsible for up to 50% security problems in codes [1]. Previous research works in this area have presented or suggested different security measures and processes towards producing secure software.

In [3], threat modeling was described as the basis for security requirements. The work mentioned that threat modeling is a process that consist three high level steps: characterizing the system, identifying assets and access points, and identifying threats. The risks associated with identified threats were then assessed.

Another tool used for identifying the security risk is the attack trees. Attack trees are used to model a chosen set of attack via a finite state machine. Attack trees model the decision making process of the attackers [2]. Attacks against a system are represented in a tree structure. The root of the tree represents the potential goal of an attacker (for example, to steal a credit card number). The nodes in the tree represent the actions the attacker can take and each path in the tree represents a unique attack to achieve the goal of the attacker. Attack trees can be used to answer question such as, what is the easiest attack, the cheapest attack, the attack that causes the most damage and so on. Attack trees are used for risk analysis, capturing of security knowledge in a reusable way and implementing counter-measures to attacks. However, the use of attack trees cannot replace the threat modeling process [5].

Another work that described the processes for producing secure software was presented at the National Cyber Summit in 2004 [4]. The work presented that security testing must be incorporated into the software development stages in order to help in producing secure software. They also explained that risk management is a tool that can help in the production of secure software. In [8], an architecture that can lead to the development of secure software products was presented. The architecture provides an efficient way of integrating security engineering into software development.

However, in most of these works, the testing and risk assessment were done using test cases derived from threat models, attack patterns, attack trees, and from specification and design. A better way of conducting the testing or evaluating the risk would be to examine the levels attained by the software for each of the software security goals.

The fuzzy logic approach has been used recently for the evaluation of risk in different situations. A fuzzy logic technique that was based on Madiami-style inference engine was used to identify the potential threats to computer-based systems [9]. The result showed an effective way of carrying out threat modeling.

Another work that was based on fuzzy modeling was presented in [6]. A cognitive fuzzy modeling technique was designed for enhanced risk assessment in a health care institution. Another similar paper was presented in [7]. The work presented a methodology for assessing and analyzing risks incurred in business information systems. In the last two papers, the risk assessment were carried out by considering what would happen if a particular decision was taken or if some information were lost or rendered unauthentic. A fuzzy cognitive map was then used to get a clear picture of the different phases at which risk can incur. The problem with this technique is that cognitive map is not computationally efficient. In this work, a rule based fuzzy expert system is used to analyze the risk associated with software before it is finally deployed.

3 DESIGN METHODOLOGY

The design is basically divided into three stages as follows:-

3.1 DESIGN OF THE LINGUISTIC VARIABLES

The inputs to the system are the values assumed or calculated for the software security goals i.e. confidentiality, integrity and availability. The goals are assumed to be of the same weight and a particular value is determined for each of them based on questions that are answered about the specific software. Also the values determined for each of the input are defined as a fuzzy number instead of crisp numbers by using suitable fuzzy sets.

Designing the fuzzy system requires that the different inputs (that is, confidentiality, integrity, and availability) are represented by fuzzy sets. The fuzzy sets are in turn represented by a membership function. The membership function used in this paper is the triangular membership function which is a three point function defined by minimum(α), maximum(β4) and modal (m) values where ($\alpha \leq \beta \leq m$) usually represented in 1.

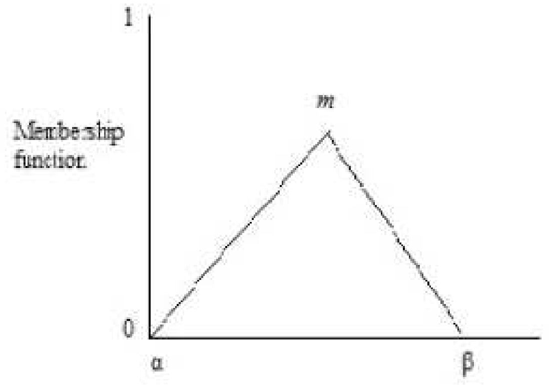


Figure 1: Triangular Membership Function

3.2 THE FUZZY SETS

The level of confidentiality is defined based on the scales of *not confidential*, *slightly confidential*, *very confidential* and *extremely confidential*. The level of integrity is also defined based on the scales *very low, low, high, very high*, and *extra high*. Also, the level of availability is also defined by the scales *very low, low, high, very high* and *extra high*. The levels defined above are based on a range definition with an assumed interval of [0 - 10]. The ranges for the inputs are shown in tables 3.2, 3.2 and 3.2.

Table 1: Range of inputs for Confidentiality

$NotConfidential$	0–1
$SlightlyConfidential$	2–3
$Confidential$	4–6
$VeryConfidential$	7–8
$ExtremelyConfidential$	9–10

Table 2: Range of inputs for Integrity

$Verylow$	Low	$High$	$VeryHigh$	$ExtraHigh$
0-1	2-3	4-6	7-8	9-10

Table 3: Range of inputs for Availabilty

$Verylow$	Low	$High$	$VeryHigh$	$ExtraHigh$
0-1	2-3	4-6	7-8	9-10

The fuzzy sets above are represented by membership functions. The corresponding membership functions for confidentiality, integrity and availability are presented in figures 2, 3 and 4 respectively.

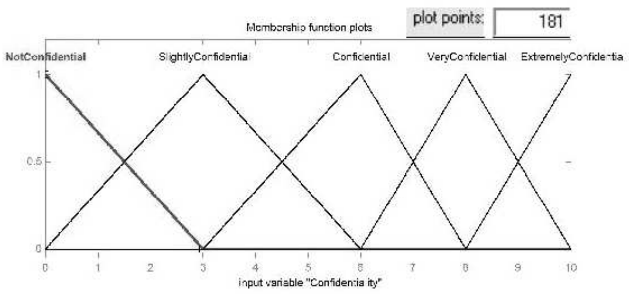


Figure 2: Membership Function for Confidentiality

Similarly, the output, that is, the level of security risk is also represented by fuzzy sets and then a membership function. The level of security risk is defined based on the scales: *not secure, slightly secure, secure, very secure*, and *extremely secure* within the range of [0 - 30]. The range definition is shown in table 3.2.

The membership function for the output fuzzy set is presented in figure 5.

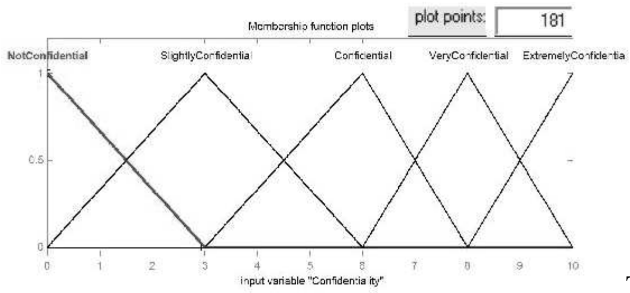


Figure 3: Membership Function for Integrity

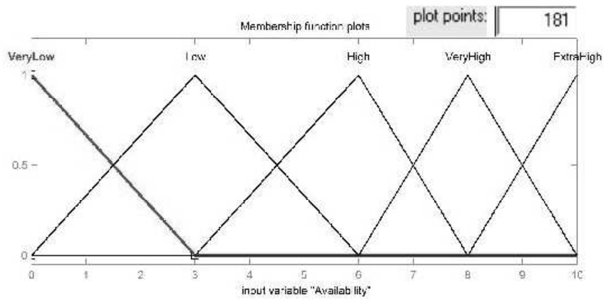


Figure 4: Membership Function for Availability

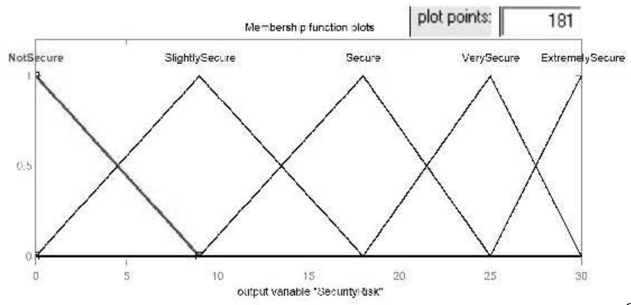


Figure 5: Membership Function for SecurityRisk

3.3 THE RULES OF THE FUZZY SYSTEM

Once the input and output fuzzy sets and membership functions are constructed, the rules are then formulated. The rules are formulated based on the input parameters (confidentiality, integrity, and availability) and the output i.e. level of security risk. The levels of confidentiality, integrity, and availability are used in the antecedent of rules and the level of security risk as the consequent of rules.

A fuzzy rule is conditional statement in the form: IF x is A

Table 4: Level of Security Risk

$NotSecure$	0–3
$SlightlySecure$	4–9
$Secure$	10–18
$VerySecure$	19–25
$ExtremelySecure$	26–30

THEN y is B

where x and y are linguistic variables; and A and B are linguistic values determined by fuzzy sets on universe of discourses X and Y, respectively. Both the antecedent and consequent of a fuzzy rule can have multiple parts. All parts of the antecedent are calculated simultaneously and resolved in a single number and the antecedent affects all parts of the consequent equally.

Some of the rules used in the design of this fuzzy system are as follow:

1. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is VeryLow) then (SecurityRisk is NotSecure).

2. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is Low) then (SecurityRisk is SlightlySecure).

3. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is High) then (SecurityRisk is SlightlySecure).

   . . .

   125. If (Confidentiality is ExtremelyConfidential) and (Integrity is ExtraHigh) and (Availability is ExtraHigh) then (SecurityRisk is ExtremelySecure)

The rules above were formulated using the Mamdani max-min fuzzy reasoning. Also, using the Mamdani max-min inference system to evaluate the complete set of rules, the security risk level derived from the rules is given in equation 1:

$$Sk_{ji}(security\, level) = max\{minC_k(x), I_j(y), A_i(z)\} \quad (1)$$

Where,

$$Ck(x) : R \rightarrow [0,1],$$
$$Ij(y) : R \rightarrow [0,1],$$
$$Ai(z) : R \rightarrow [0,1].$$

The equation 1 is the fuzzy reasoning for evaluating a rule base to get the output i.e. the defuzzification procedure. The procedure used here is the Center Of Area (COA).

The rules formulated are then transformed to fuzzy perception structure. The fuzzy perception is normally determined by the rules and fuzzy sets of the underlying problem. The fuzzy perception describes the relationships between the variables during and after evaluation of the output. A reduced form of the fuzzy perception representation of the rules is given in figure 6.

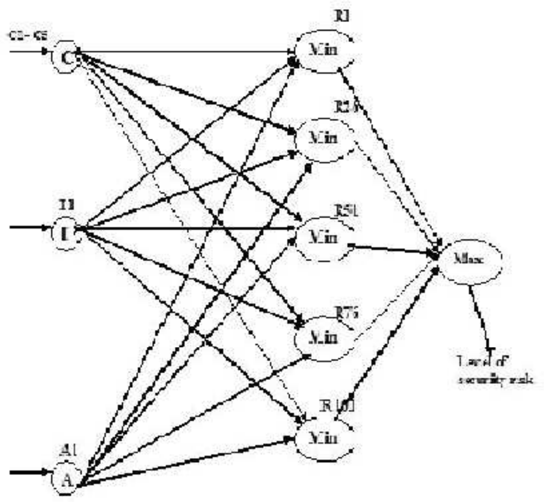


Figure 6: Fuzzy perception structure

A summary of the procedure with which the initial fuzzy inference system is built is presented below.

3.4 Procedure for building the initial fuzzy inference system

Step 1: Determining the input variables.
The input variables are determined based on the properties that should be incorporated or eliminated in a system. For example the properties to be incorporated into a secure system are used in this work. These are confidentiality, integrity and availability.

Step 2: Defining the input variables.
For each of the input, suppose that the domain interval is $[0, +s]$ e.g. $[0 - 10]$, each of them defined individually. The domain is divided into 2N + 1 regions and each region is attached a fuzzy membership function. In this work, the domain is divided into 5 regions (N = 2). The regions are represented by triangular membership functions.

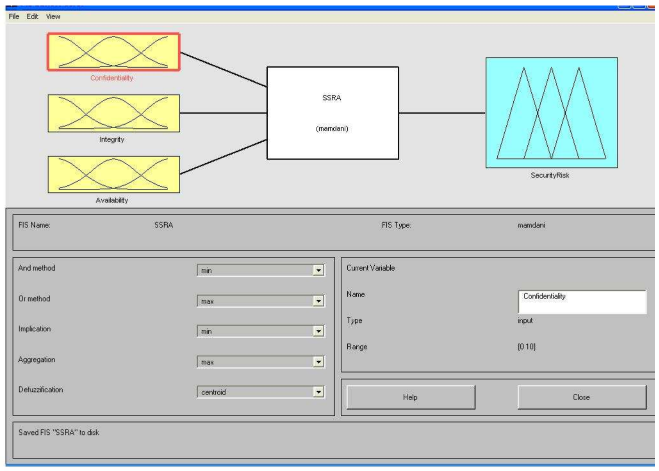
Step 3: Defining the output variable membership function.
Suppose the domain interval for the output is $[0, +s]$ e.g. $[0 - 30]$. Divide the domain interval into 2N + 1 region and assume for each region a membership. For example level of security risk (the output) is divided into 5 regions (N = 2).

Step 4: Formulating the rules and populating the rule base.
The rules are built based on expert knowledge of the relationships between the variables. The rules are formulated to reflect the relationships between any possible relation of the input variables and the output variable. The rules in this work reflect the relationships between the levels of confidentiality, integrity and availability and also the level of security risk. Thus, there are $(2N + 1)^3$ fuzzy rules in the rule base of the fuzzy system.

4 IMPLEMENTATION PROCEDURE

The linguistic variables were determined with the extent of the positive and negative responses to a well-constructed security questions that are presented in form of on-line questionnaire. As it was mentioned earlier, MATLAB was used for the implementation. The linguistic inputs to the system are supplied through the graphical user interface called rule viewer. Once the rule viewer has been opened, the input variables are supplied in the text box captioned *input* with each of them separated with a space.

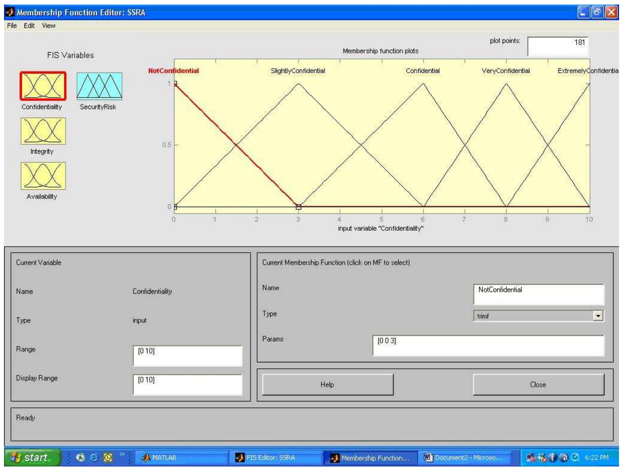
a. THE FIS EDITOR
The fuzzy inference system editor (7) shows a summary of the fuzzy inference system. It shows the mapping of the inputs to the system type and to the output. The names of the input variables and the processing methods for the FIS can be changed through the FIS editor.



FIS Name: SSRA
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: Confidentiality
Type: input
Range: [0 10]
Saved FIS "SSRA" to disk

Figure 7: The FIS editor

b. THE MEMBERSHIP FUNCTION EDITOR
This can be opened from the command window by using the *plotmf* function but more easily through the GUI. The membership function editor (8) shows a plot of highlighted input or output variable along their possible ranges and against the probability of occurrence. The name and the range of a membership value can be changed, so also the range of the particular variable itself through the membership function editor.



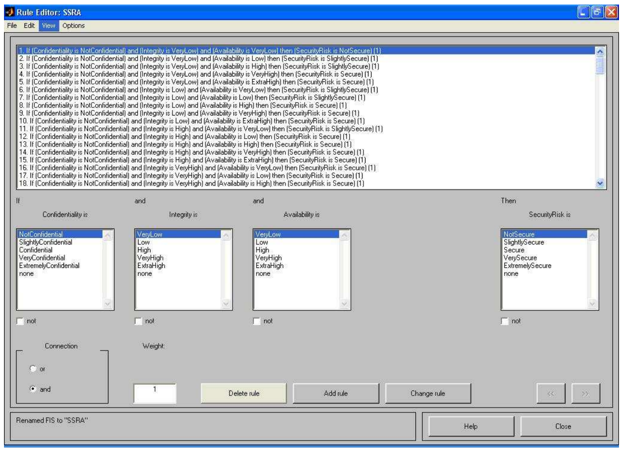
```
Current Variable
Name            Confidentiality
Type            input
Range           [0 10]
Display Range   [0 10]

Current Membership Function (click on MF to select)
Name            NotConfidential
Type            trimf
Params          [0 0 3]

Ready
```

Figure 8: The Membership Function editor

c. THE RULE EDITOR
The rule editor can be used to add, delete or change a rule. It is also used to change the connection type and the weight of a rule. The rule editor for this application is shown in figure 9.



```
1. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is VeryLow) then (SecurityRisk is NotSecure) (1)
2. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is Low) then (SecurityRisk is SlightlySecure) (1)
3. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is High) then (SecurityRisk is SlightlySecure) (1)
4. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is VeryHigh) then (SecurityRisk is Secure) (1)
5. If (Confidentiality is NotConfidential) and (Integrity is VeryLow) and (Availability is ExtraHigh) then (SecurityRisk is Secure) (1)
6. If (Confidentiality is NotConfidential) and (Integrity is Low) and (Availability is VeryLow) then (SecurityRisk is SlightlySecure) (1)
7. If (Confidentiality is NotConfidential) and (Integrity is Low) and (Availability is Low) then (SecurityRisk is SlightlySecure) (1)
8. If (Confidentiality is NotConfidential) and (Integrity is Low) and (Availability is High) then (SecurityRisk is Secure) (1)
9. If (Confidentiality is NotConfidential) and (Integrity is Low) and (Availability is VeryHigh) then (SecurityRisk is Secure) (1)
10. If (Confidentiality is NotConfidential) and (Integrity is Low) and (Availability is ExtraHigh) then (SecurityRisk is Secure) (1)
11. If (Confidentiality is NotConfidential) and (Integrity is High) and (Availability is VeryLow) then (SecurityRisk is SlightlySecure) (1)
12. If (Confidentiality is NotConfidential) and (Integrity is High) and (Availability is Low) then (SecurityRisk is Secure) (1)
13. If (Confidentiality is NotConfidential) and (Integrity is High) and (Availability is High) then (SecurityRisk is Secure) (1)
14. If (Confidentiality is NotConfidential) and (Integrity is High) and (Availability is VeryHigh) then (SecurityRisk is Secure) (1)
15. If (Confidentiality is NotConfidential) and (Integrity is High) and (Availability is ExtraHigh) then (SecurityRisk is Secure) (1)
16. If (Confidentiality is NotConfidential) and (Integrity is VeryHigh) and (Availability is VeryLow) then (SecurityRisk is Secure) (1)
17. If (Confidentiality is NotConfidential) and (Integrity is VeryHigh) and (Availability is Low) then (SecurityRisk is Secure) (1)
18. If (Confidentiality is NotConfidential) and (Integrity is VeryHigh) and (Availability is High) then (SecurityRisk is Secure) (1)

Renamed FIS to "SSRA"
```

Figure 9: The Rule editor

d. THE RULE VIEWER
The text box captioned input is used to supply the three input variables needed in the system. The appropriate input corresponds to the number of *YES* answer in the questionnaire for each of the input variables. For example, in the figure 10, all the input variables are 5 and the corresponding output is 13.9, which specified at the top of the corresponding graphs. The input for each of the input variables is specified at the top of the section corresponding to them, so also the output variable. The rule viewer for this work is presented in figure 10.

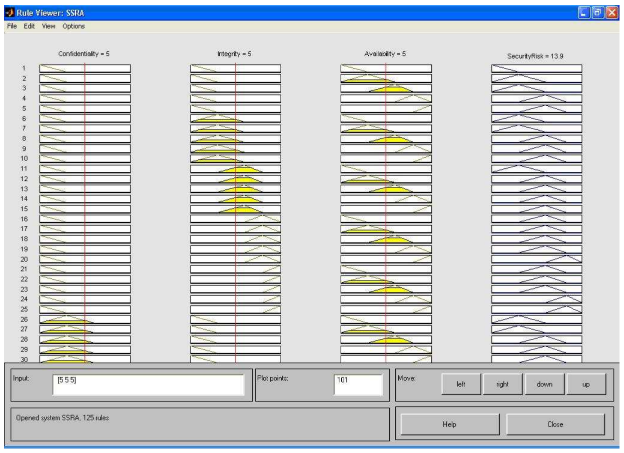


Figure 10: The Rule editor

e. THE SURFACE VIEWER
The surface viewer shown in figure 11 is a 3-D graph that shows the relationship between the inputs and the output. The output (securityRisk) is represented on the Z-axis while 2 of the inputs (Confidentiality and Integrity) are on the x and y axes and the other input (Availability) is held constant. The surface viewer shows a plot of the possible ranges of the input variables against the possible ranges of the output.

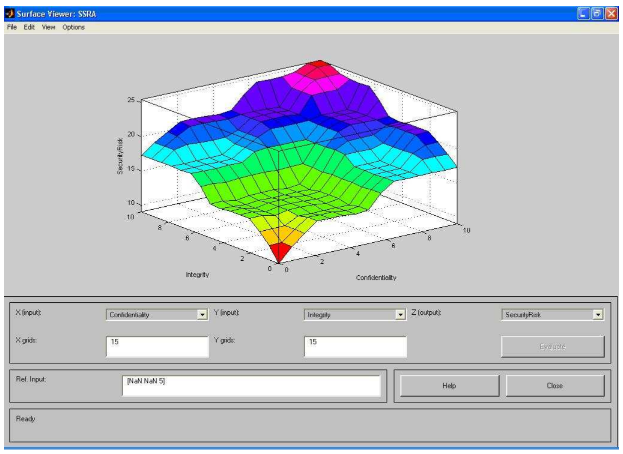
5 EVALUATION

The security risk analysis system was evaluated using three newly completed software products from three different software development organizations. The output determines the security level of software under consideration. The summary of the evaluation is given in Table 5.

For product A, 5 is the score for confidentiality, 5 for the integrity and 5 for the availability. The crisp output 13.9 shows that the security level for the soft-

$Softare$	$InputvariablesA$	$CrispoutputC$	$Significance$	$SecurityLevel$
A	5 5 5	13.9	45% slightly secure, 55% secure	46.33%
B	8 7 8	24.2	20% secure, 80% very secure	80.60%
C	10 10 10	28.4	35% very secure, 65% extremely secure	94.67%

Table 5: Evaluation of Different Input Variables



X (input): Confidentiality
Y (input): Integrity
Z (output): SecurityRisk
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 5]
Ready

Figure 11: The Surface Viewer

ware is 13.9 out of the 30. On the fuzzy sets defined for the security risk level, this value corresponds to around 45% slightly secure and 55% secure. On the scale of 30, this value shows that the software having the values is 46.33% secure. In an organization where the security level has been set for like a minimum 70%, it is evident that this software is not secure and necessary adjustments have to be made. In order to make these adjustments, the software has to be redesigned or examined in order to improve on the security. For product B, the inputs to the system are 8, 7 and 8 for confidentiality, integrity and availability respectively. The crisp output is 24.2. This value corresponds to 20% secure and 80% very secure on the fuzzy set scale for the output. This then means that the software is 80.67% secure. Thus for an organization whose standard is a minimum of 70% software security, it can be concluded that the software is secure enough and can then be deployed.

From the last product C, it was seen that a software can never be 100% secure which is actually the case because no one can be perfectly sure that the software is error free. When the inputs to the system are 10, 10 and 10 for confidentiality, integrity and availability respectively (which are the highest possible values from the design), the output to the system is 28.4 on the output scale, which corresponds to around 35% very secure and 65% extremely secure. On the general scale, this value means that the average level for which one can claim the security of any software is around 94.67% and it rare to claim that a software is 100% secure.

6 FUTURE WORK AND CONCLUSION

It might be necessary to redesign this system in a way that it will be deployable and will be without the use of MATLAB. It might also be necessary to use an adaptive fuzzy logic technique for security risk analysis.

We have been able to design a system that can be used to evaluate the security risk associated with the production of secure software systems. This will definitely help software organizations meet up with the standard requirements. A technique for assessing security of software system before final deployment has been presented.

The result of this study shows that if the software producing companies will incorporate security risk analysis into the production of software system, the issue of insecurity of software will be held to the minimum if not eliminated. This study has also revealed that if each of the software security goals can be increased to the maximum, then the level security will also be increased and the risk associated will be eliminated.

Finally, security risk analysis is a path towards producing secure software and should be considered a significant activity by software development organisations.

References

[1] Hoglund, G. and McGraw, G. Exploiting software, how to break the code. *Addison-Wesley publisher)*, 2004.

[2] Leveson, N. G. System safety and computers. *Addison-Wesley publisher*, 1995.

[3] Myagmar, A. J., S.and Lee and Yurcik, W. Threat modeling as basis for security requirements. *In Symposium on Requirements Engineeing for Information Security (SREIS)*, 2005.

[4] Noopur, D. and Samuel, T. R. Processes to produce secure software. *A paper written for, Cyber*

Security Summit Taskforce Subgroup on Software Development Lifecycle, 2004.

[5] Sheyner, O. and Wing, J. Tools for generating and analyzing attack graphs. *In Proceedings of formal methods for component and Objects*, 2005.

[6] Smith, E. and Eloff, J. Cognitive fuzzy modeling for enhanced risk assessment in a health care institution. *IEEE Intelligent systems & their applications*, 15(2), 2000.

[7] Smith, E. and Eloff, J. Transaction based risk analysis using cognitive fuzzy techniques. *Advances in Information Security Management & Small Systems Security*, 2001.

[8] Sodiya, A. S., Onashoga, A., S., and Oladunjoye, B. A. Threat modeling using fuzzy logic paradigm. *Journal of Issues in Informing Science and Information Technology. U. S. A*, 15(4):53–61, 2007.

[9] Sodiya, A. S., Onashoga, S. A., and Oladunjoye, O. B. Towards building secure software products. *Journal of Issues in Informing Science and Information Technology, U. S. A*, 13:635–646, 2006.

[10] Zadeh, L. A. Fuzzy sets. *Information and Control*, 1965.