Threat Modeling Using Fuzzy Logic Paradigm

A. S. Sodiya, S. A. Onashoga, and B. A. Oladunjoye
Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

sinaronke@yahoo.co.uk; bookyy2k@yahoo.com; oladunjoyeak@yahoo.com

Abstract

Threat modeling, which is the process of identifying, quantifying and analyzing potential threats of computer-based systems, has become a significant consideration towards designing secure software systems. Despite the previous methods adopted for threat modeling, there are still many systems that are probable to attack. In this work, a fuzzy logic-based threat modeling technique is designed. The technique involves the fuzzification of input variables that is based on six major categories of threats (STRIDE- Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege), rule evaluation, and aggregation of the rule outputs. The design is based on Mamdani-style inference system which is very good for the representation of human reasoning and effective analysis. The implementation is done using MATLAB fuzzy logic tools. Using the design to test five computer systems, the result shows a tool that can be effectively used to analyze potential threats to computer-based systems.

Keywords: Fuzzy logic, threat, threat modelling

Introduction

Nowadays, risk analysis plays a significant role in security management efforts. It was explained in Davis et al., 2004 that risk management is a tool that can help in production of secure software. In other words, security risk analysis is crucial to producing secure software products.

It is widely accepted that designing secure computer system is a difficult problem. Attackers frequently break into systems and cause a lot of destruction and as a result, software users are now interested in secure software products. In the past, many techniques have been developed to solve a wide array of security problems. Poorly chosen security measures can prove to be useless, or even counterproductive in the face of a well-executed attack. As such, the security system designers must be interested in incorporating security into every stage of software design.

Also, in order to adequately protect computer systems, it is important to properly understand the system and its associated potential threat. There are several processes for identifying and prioritizing risk. One of the most effective is threat modeling. Threat modeling is the process of identifying, quantifying and analyzing potential threats of a computer-based system. It is a process of assessing and documenting a system's security risks (Ambler, 2005). It involves identifying the key assets of an application, decomposing the application, identifying and categorizing the threats to each assets or component, rating the threats based on a risk ranking, and then developing threat mitigation strategies that are then implemented in design, code and test cases (Davis et al., 2004).

Categorizing threats is the first step toward effectives mitigation (Ambler, 2005). Threats can be classified into six classes based on their effect (Swiderski et al., 2004):

1. Spoofing: - Using someone else credentials to gain access to otherwise inaccessible assets. A process mounts a spoofing attack by passing forged or stolen credentials.
2. Tampering: - Changing data to mount an attack.
3. Repudiation: - Occurs when a user denies performing an action, but the target of the action has no way to prove otherwise.
4. Information disclosure: - The disclosure of information to a user who does not have permission to see it.
5. Denial of service: - threatens the ability of valid users to access resources.
6. Elevation of privilege: - Occurs when an unprivileged user gain privileged status.

This is generally referred to as the STRIDE model. The STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, and Elevation of privilege) model was used by Microsoft for categorizing threats (Casteele, 2005). The model is used by developers and designers to identify and resolve security issues in the application code. This means that the STRIDE model is used to categorize the threats by taking into account their effects on the security of the application (Casteele, 2005).

The process of threat modeling helps system architects assess and document the security risks associated with a system. Identifying threats helps to develop efficient, realistic and meaningful security requirements for computer-based systems. This is particularly important because if the security requirements are faulty, then the system cannot be secure. Proper identification of threats and appropriate selection of countermeasures helps reduce the ability of attackers to misuse the system. Rigorous analysis of security requirements can detect security design flaws, allowing their correction prior to costly development and deployment of flawed systems. In this work, a fuzzy logic-based technique is designed for effective threat modeling.

The rest of this work is organized as follows. Section 2 present related works. In section 3, the methodology of the design is discussed. Section 4 presents the implementation and evaluation procedure. Future work and conclusion are presented in section 5.

Literature Review

Threat modeling is a sound approach to addressing software risks at the design level (Hoglund, 2004). Based on the decomposition of application, it evaluates the threats and risks to a system and chooses techniques to mitigate the threats. Security threats are modeled by attack trees, which describe the decision – making process attackers would go through to compromise the system.

Casteele (2005) presented a paper on threat modeling for web applications. This work focused on the important OWASP (Open Web Application Security Portal) Top 10 Web application security vulnerabilities/ problems. An architecture that is based on STRIDE model was used for the analysis. Klein (2005) presented a similar technique of carrying out threat modeling on a voting system.

Several academic teams jointly wrote a paper on analysis of threats that occurred when smart cards are used in web applications (De Cock et al., 2004). Their analysis was a part of the Designing Secure Applications (DeSecA) project, funded by Microsoft. The aim of their project was to provide an application developer with a tool that allows him to prevent the exploitation of a broad range of threats. They investigated common threats in five areas, each focusing on one particular technological building block for web applications. One of these was the smart card, and in particular the electronic identity card (De Cock et al., 2004).

Sodiya et al. (2006) presented an architecture for producing secure software and threat modeling was the foundation of this architecture. A comprehensive internet browser threat model was discussed in Griggs, 2004. He first listed various well known threat models on the internet and then identified the ones specific to browsers. The main types of attacks discussed are phishing, eavesdropping, man-in- the- middle, and denial of service (DoS).

Most of the papers discussed threat modeling for applications and how to incorporate it into development process of application products (quantitative). Our work differs from them in that it provides a systematic approach(qualitative) to threat modeling for computer-based systems using six categories of potential threats (STRIDE model).

Design Methodology

The system is designed using fuzzy inference system which is a popular computing framework based on the concept of fuzzy set theory, fuzzy if – then rules, and fuzzy reasoning.

The Model

The fuzzy logic-based threat modeling architecture is given in Figure 1.

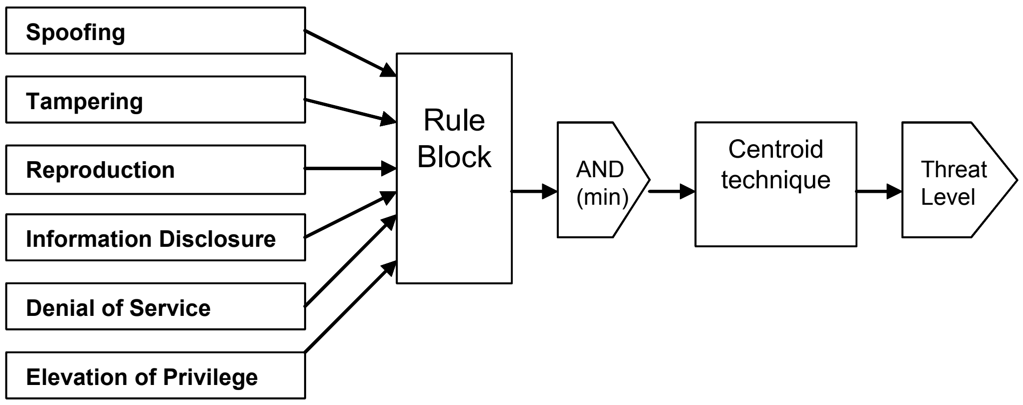


Figure 1: Architecture for Fuzzy Logic-based threat modelling

The steps involved in the design are:

Fuzzification → Rule Evaluation→ Aggregation→Defuzzification →Output

a. Fuzzification: This is the process of generating membership values for a fuzzy variable using membership functions. The first step is to take the crisp inputs (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege) and determine the degree to which these inputs belong to each appropriate fuzzy set. This crisp input is always a numeric value limited to the universe of discourse. Once the crisp inputs are obtained, they are fuzzified against the appropriate linguistic fuzzy sets.

Membership function is designed for each potential threat which is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between

[0, 1]. Linguistic values are assigned for each threat as Low, Moderate, and High while for threat level as Very low, Low, Rather low, Medium Rather high, High, and Very high. For each input their values ranges from 0 to 10 while for output, ranges from 0 to 100.

The following tables contain the linguistic variables and their ranges

1. **Linguistic variable: Spoofing, S**

Linguistic value	Numerical range
Low	[0, 0, 3, 6]
Moderate	[4, 6, 8]
High	[6, 8, 10, 10]

2. **Linguistic variable: Tampering, T**

Linguistic value	Numerical range
Low	[0, 0, 2.5, 5]
Moderate	[4, 6, 8]
High	[7, 8.5, 10, 10]

3. **Linguistic variable: Repudiation, R**

Linguistic value	Numerical range
Low	[0, 0, 2, 5]
Moderate	[3.5, 6, 7.5]
High	[6, 8, 10, 10]

4. **Linguistic variable: Information Disclosure, I**

Linguistic value	Numerical range
Low	[0, 0, 2, 4.5]
Moderate	[3.5, 5.5, 7.5]
High	[6.5, 8, 10, 10]

5. **Linguistic variable: Denial of Service, D**

Linguistic value	Numerical range
Low	[0, 0, 2.5, 5.5]
Moderate	[4, 6, 8]
High	[6.5, 9, 10, 10]

6. **Linguistic variable: Elevation of Privilege, E**

Linguistic value	Numerical range
Low	[0, 0, 2.5, 6]
Moderate	[4, 6, 8]
High	[6, 8.5, 10, 10

7. **Linguistic variable: Threat level**

Linguistic value	Numerical range
Very low	[0, 0, 30]
Low	[0, 20, 40]
Rather low	[25, 35, 45]
Medium	[40, 50, 70]
Rather high	[55, 65, 75]
High	[60, 80, 100]
Very high	[75, 100, 100]

b. Rule Evaluation: This is the second step where the fuzzified inputs are applied to the antecedents of the fuzzy rules. Since the fuzzy rule has multiple antecedents, fuzzy operator (AND or OR) is used to obtain a single number that represents the result of the antecedent evaluation. We apply the AND fuzzy operation (intersection) to evaluate the conjunction of the rule antecedents. Rules added to this system are about 139 which were derived by mapping the six inputs to one output by using conjunction (AND).

c. Aggregation of the rule outputs: This is the process of unification of the outputs of all rules. In other words, we take the membership functions of all the rules consequents previously scaled and combine them into single fuzzy sets (output). Thus, input of the aggregation process is the list of scaled consequent membership functions, and the output is one fuzzy set for each output variable.

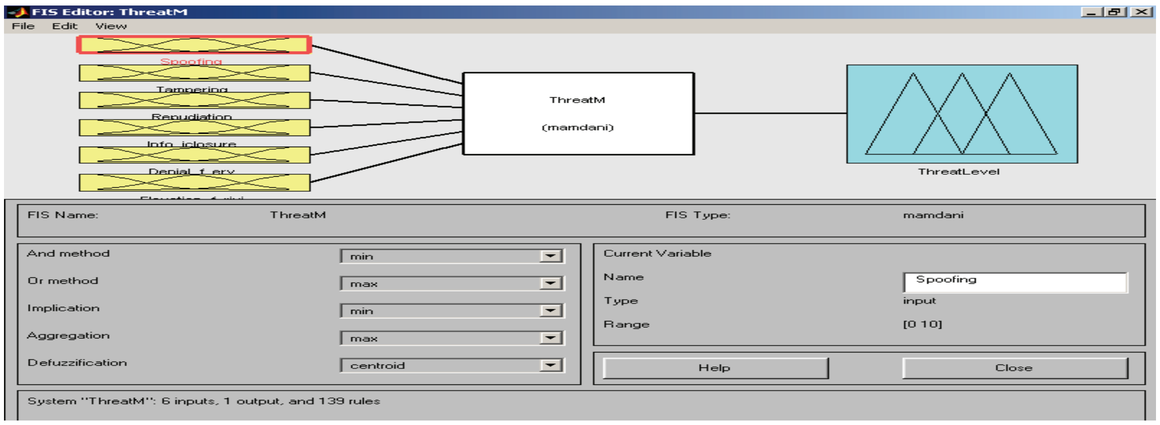
d. Defuzzification: This is the last step in the fuzzy inference process, which is the process of transforming a fuzzy output of a fuzzy inference system into a crisp output. Fuzziness helps to evaluate the rules, but the final output this system has to be a crisp number. The input for the defuzzification process is the aggregate output fuzzy set and the out put is a number. This step was done using Centroid technique because it is most commonly used method of defuzzification.

Implementation and Evaluation

Implementation

The design is implemented using MATLAB fuzzy logic toolbox. The interfaces of the implementation are presented below:-

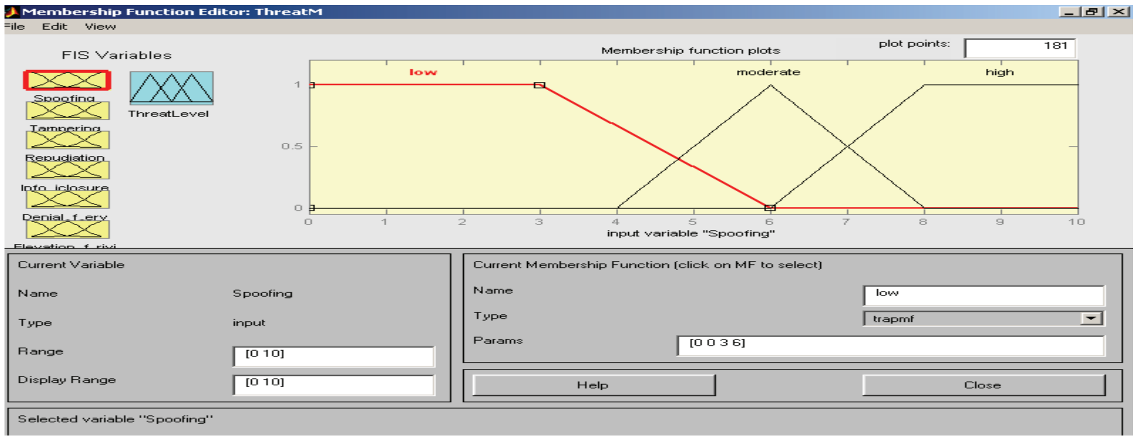
a. **FIS Editor** (Figure 2): This is the window through which a new FIS type with any particular model can be selected, variable can be added, and input or output variable names can be changed. In this case, the chosen model is Mamdani.



FIS Name: ThreatM
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: Spoofing
Type: input
Range: [0 10]
System "ThreatM": 6 inputs, 1 output, and 139 rules

Figure 2: Fuzzy inference Editor

b. **Membership function editor** (Figures 3 and 4): This is the window through which the input or the output of the membership function can be changed and membership function can be added or removed. It also makes it possible to specify the ranges of each of the variables and membership functions.



plot points: 181

Current Variable

Name Spoofing

Type input

Range [0 10]

Display Range [0 10]

Current Membership Function (click on MF to select)

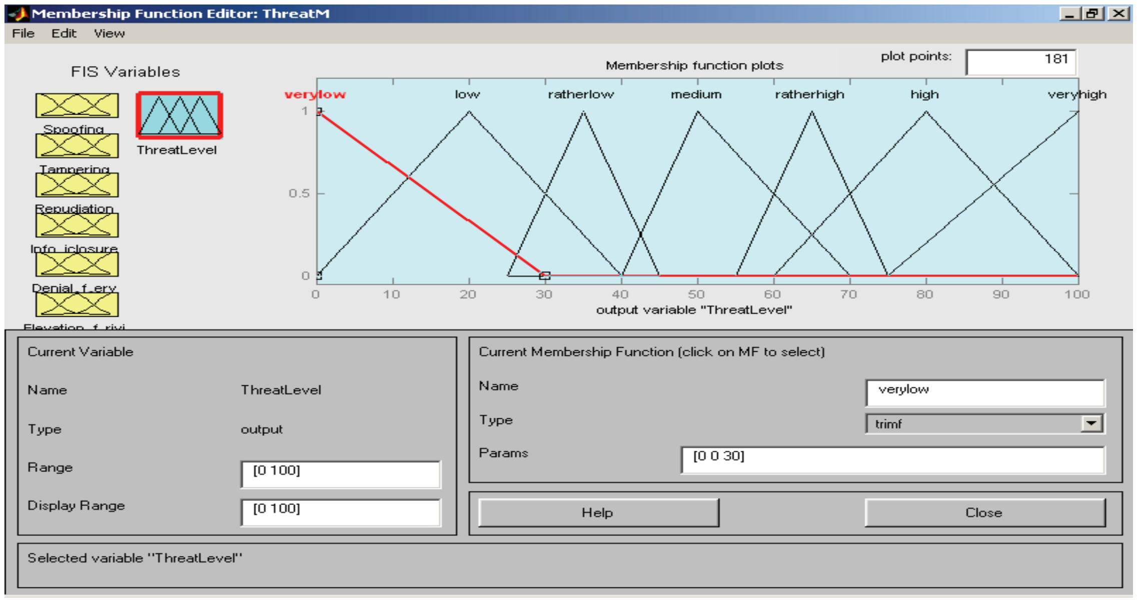
Name low

Type trapmf

Params [0 0 3 6]

Selected variable "Spoofing"

Figure 3: Membership function specification for Spoofing



plot points: 181

Current Variable

Name ThreatLevel

Type output

Range [0 100]

Display Range [0 100]

Current Membership Function (click on MF to select)

Name verylow

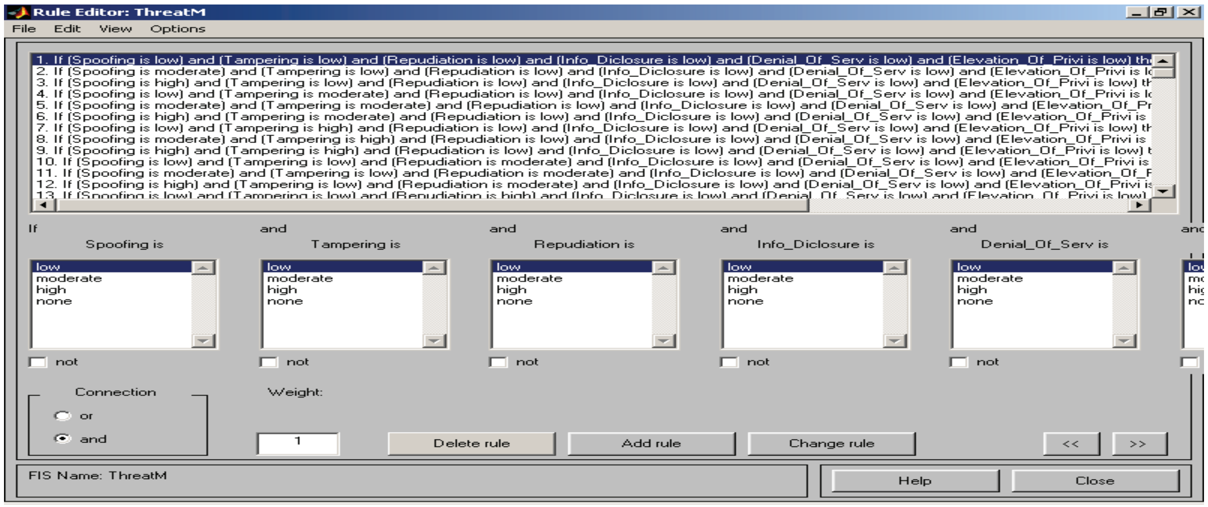
Type trimf

Params [0 0 30]

Selected variable "ThreatLevel"

Figure 4: Threat Level (Output) membership function

c. <u>Rule editor</u> (Figure 5)**:** This is used to add, change or delete rules, as the name implies. It provides opportunity to change the connections and weight applied to the rules (the default weight always is 1).



```
1. If (Spoofing is low) and (Tampering is low) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is low) th
2. If (Spoofing is moderate) and (Tampering is low) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is lo
3. If (Spoofing is high) and (Tampering is low) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is low) th
4. If (Spoofing is low) and (Tampering is moderate) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is lo
5. If (Spoofing is moderate) and (Tampering is moderate) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Pr
6. If (Spoofing is high) and (Tampering is moderate) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is
7. If (Spoofing is low) and (Tampering is high) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is low) th
8. If (Spoofing is moderate) and (Tampering is high) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is
9. If (Spoofing is high) and (Tampering is high) and (Repudiation is low) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is low) t
10. If (Spoofing is low) and (Tampering is low) and (Repudiation is moderate) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is
11. If (Spoofing is moderate) and (Tampering is low) and (Repudiation is moderate) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_F
12. If (Spoofing is high) and (Tampering is low) and (Repudiation is moderate) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is
13. If (Spoofing is low) and (Tampering is low) and (Repudiation is high) and (Info_Diclosure is low) and (Denial_Of_Serv is low) and (Elevation_Of_Privi is low)
```

Rule Editor: ThreatM
File Edit View Options
If Spoofing is: low, moderate, high, none
and Tampering is: low, moderate, high, none
and Repudiation is: low, moderate, high, none
and Info_Diclosure is: low, moderate, high, none
and Denial_Of_Serv is: low, moderate, high, none
not
Connection: or, and
Weight: 1
Delete rule
Add rule
Change rule
<<
>>
FIS Name: ThreatM
Help
Close

Figure 5: Rules Editor

d. <u>Rule Viewer</u> (Figure 6)**:** The rule viewer shows a graphical representation of each of the variables through all the rules, a representation of the combination of the rules, and a representation of the output from the defuzzification. It also shows the crisp value output of the system. Data are entered for analysis through the Rule Viewer at the Input text field.

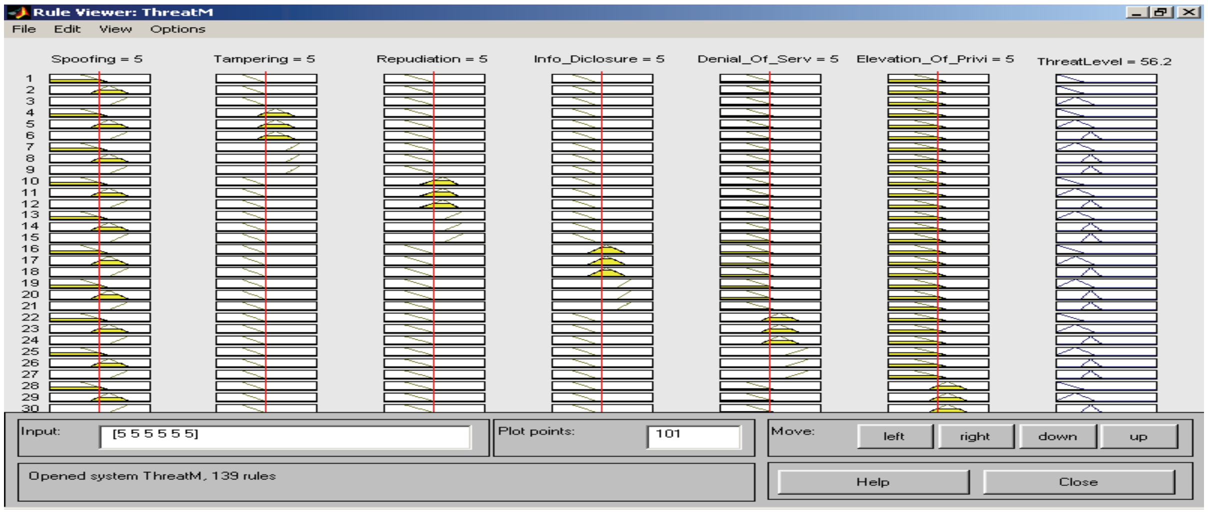


Figure 6: Rule viewer

Evaluation

The evaluation of the systems shows a system that can be effectively used for qualitative risk assessment. The computer based systems were used to analyze using different assumed values and the output shows that the system is robust enough for the evaluation of the threat level. The output measures the level of an event or activity with the potential to cause harm to their systems. The output is can be a value from 0 to 100 each corresponding to the level of threat.

Five computer systems were analyzed using this system to determine the threat level. Results are shown in Table 1.

Table 1: Results of the threat level on five computer systems

Systems	Threat level	Implication
A	56.2	Medium
B	65	Rather high
C	18.5	Low
D	90.5	Very high
E	65	Rather high

Conclusion and Future Work

Conclusion

In this work, fuzzy based system was designed to evaluate the threat level of identified threats, because it is impossible to provide assurance for the system and justify security measures incorporated unless the system is analyzed during the designing state of computer based systems. With this system designed, risk analysis has been made easer to perform.

Future Work

For further research, this system designed can be redesigned using object orientated programming language and other models like DREAD and SWOT model can be used.

References

Ambler, W. S (2005). Introduction to security threat modeling. Agile Modeling. Available at http://www.agilemodeling.com/artifacts/securityThreatModel.htm

Casteele, S.V. (2005). Threat modeling for web application using STRIDE model.

Davis, N., Howard, M., Humphrey, W., McGraw, G., Redwine Jr., S. T., Zibulski, G., & Graettinger, C. (2004). Processes to produce secure software: Towards more secure software. A report at National Cyber Security Summit, Vol. 1. Available at http://www.cigital.com/papers/download/secure_software_process.pdf

De Cock, D., Wouters, K., Schellekens, D., Singelee, D., & Preneel. B. (2005). Threat modeling for security tokens in web applications. In *Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security* (CMS '04) September 2004, pp 183-193.

Griggs, I. (2004). Browser threat model. Retrieved from http://iang.org/ssl/browser threat model.html

Hoglund, G. & McGraw, G. (2004). *Exploiting software: How to break code.* Addison –Wesley Professional.

Klein, S. A. (2005). Position paper on voting system threat modeling. Available at http://vote.nist.gov/threats/papers/threat-modelling.pdf

Sodiya, A.S, Onashoga, S.A, & Ajayi, O.B (2006). Toward Building Secure Software Products. *Journal of issues in Informing Science and Information Technology, 3,* 635-646. Available at http://informingscience.org/proceedings/InSITE2006/IISITSodi143.pdf

Swiderski, F. & Snyder, W. (2004). *Threat modeling*. Microsoft Press Professional Book Series

Biographies

Dr. **Adesina Simon Sodiya** is presently a lecturer in the department of Computer Science, University of Agriculture, Abeokuta. He has published in both local and international journals. His research areas are Computer Security, Artificial Intelligence, Software Engineering, Data mining.

Onashoga, Saidat Adebukola (nee OKUNLAYA, formerly IBRAHIM, S. A.) has worked as a lecturer in the Department of Computer Science, University of Agriculture, Abeokuta, Nigeria for the past six years. She had her degrees in University of Agriculture, Abeokuta, Nigeria with M.Sc. in Computer Science (2004), B.Sc. Mathematical Science (Computer Science option, 2000). She is currently on her Ph.D. programme in Computer Science in the same University. She has published in both International and local journals. Her current research interests include Network Security, Data Mining and Knowledge Management. She is married.

B. A. Oladunjoye [Biography not available]