J. Basic. Appl. Sci. Res., 3(3)1123-1128, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

FDMSWAP: Formal Development Methodology for Secure Web Applications

Shafiq Hussain[1], Ghulam Rasool[2], Muhammad Atef[2], Abdul Karim Shahid[2]

[1]Department of Computing, Engineering and Technology, University of Sunderland, Sunderland, United Kingdom
[2]Department of Computer Science, COMSATS Institute of IT, Lahore, Pakistan

ABSTRACT

Computer security is beyond the external protections of software systems. Main concern lies inside the software systems due to weak and non-verified design. A significant portion of the software security problems lies within the software itself. Internal software security is more critical as compared to external protections. Many techniques and methodologies have been developed to address the issues of internal security of software systems. Threat modeling is one of such techniques. Threat modeling is a methodology performed at the early stages of software development. In threat modeling all possible threats to software systems can be identified. Also mitigation measures are generated as part of this process. Threat modeling methodologies such as Secure Development Lifecycle (SDL) and Threat Analysis Methodology (TAM) practicing in industry and academia rely on informal and semi-formal techniques. These techniques lead to inconsistencies in the design of systems due to ambiguities of these techniques. Formal methods are based on mathematics and used for specification, analysis and design of software systems. By using formal methods software systems can be analyzed and verified at the design level. As a result of this, more accurate, reliable and inconsistencies free software systems can be produced. In this paper, a new methodology based on formal methods has been introduced for developing secure web applications. The central idea of this methodology: Formal Development Methodology for Secure Web Applications (FDMSWAP) is threat modeling process to extract security requirements at the early stages of software development and to integrate these security properties into the design of software. Then the resulting design is analyzed, verified and validated by using formal methods Event-B and RODIN.
KEYWORDS: Threat Modeling, Formal Methods, Proof Obligations, Security, Event-B, RODIN, STRIDE, DREAD.

1. INTRODUCTION

Due to the rapid magnification of distributed systems, chances of security breaches also become higher and software applications are more vulnerably susceptible to attacks due to higher skills of attackers [7]. The increasing dependence of devices on computers, interlinked with networks and internet has made security a fundamental aspect of systems [18]. In the past security was considered as an additional activity instead of considering as a component of development process and all attention was on the functional aspects of systems. In addition, security considerations were on the protective mechanisms instead of dealing in a similar way as with functional requisites. Security properties must be integrated into the systems at the design level [17] and [8]. Many methodologies have been developed for this purpose. The latest developments in this regard were to consider threat modeling as part of secure development process. Threat modeling is a methodology used to identify all possible threats to software systems. A threat is a possible harm that may be occurred due to vulnerability present in the design of software applications [19]. By performing threat modeling at the early stages of development, all the possible threats can be identified along with mitigation measures. This paper describes a methodology predicated on formal methods for the development of secure web applications. The presented methodology starts from requirements specifications and leads towards a ready to implement formal model of applications. The rest of the paper is as follows: Section 2 describes related work. Section 3 introduces formal methods. Section 4 presents a summary of security properties. In section 5, basic security models are introduced. Section 6 discusses the methodology. Conclusion is presented in section 7.

2. Related Work

STRIDE is most widely used threat model and is being utilized in the tools such as SDL and TAM [13]. STRIDE covers the main categories of threats such as Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege[15]. After the threats are identified, these threats are rated by the DREAD model. DREAD includes Damage potential, Reproducibility, Exploitability, Affected uses and Discoverability [21]. Abuser stories are the extensions of agile practices used to identify security imperfections in the system by identifying how the attackers may abuse the system [22]. In Agile modeling, threats modeling process is predicated on five elements: identifying threats, understanding threats, categorizing threats utilizing STRIDE, identifying mitigation strategies and testing [16]. STRIDE Average Model is an enhanced model of the STRIDE model. In this model software applications are divided into five categories and are rated according to the category of the application. Then information criticality is determined depending on the information, the software application will deal. Required controls are identified and STRIDE average is calculated. At the end potential security risk is calculated. The approach did not have support of any automated tool[11].Threat modeling utilizing attack trees is another methodology used to model threats to software systems. In this methodology, a structure similar to a tree is developed to model attacks on the system resources. In attack tree methodology, node of the tree represents a goal and leafs of the tree represents different ways to reach the goal. After the tree is built different values can be assigned to the leaf nodes and these values are utilized to calculate the security of the goal. Other attributes such as time required to complete a step and operational expenses etc can also be integrated to trees [24]. Secure ITree is a graphical tool to model attack trees by Amenaz

Corresponding Author: Shafiq Hussain, Department of Computing, Engineering and Technology, University of Sunderland, Sunderland, United Kingdom. Tel: +447466638765 Email: shafiq.hussain@research.sunderland.ac.uk

Technologies and it is based on mathematical attack tree model. Secure ITree has been used successfully in many applications such as buildings, pipelines and electrical transmission lines. Secure ITree can be used to identify vulnerabilities in software applications [20]. The fuzzy logic-predicated threat modeling technique is predicated on the fuzzy set theory. In this technique, input variables predicated on STRIDE model are passed to fuzzy inference engine and associated threats are identified as output. MATLAB fuzzy logic toolset is utilized to automate the process [23]. SDL Threat Modeling Tool and Application Threat Modeling (TAM) Tool developed by Microsoft are predicated on STRIDE model and DREAD model. In SDL and TAM, a model of proposed system is given as DFD as input and these tools generate reports describing all the possible threats to the system along with mitigations to reported threats [6] and [14].T-MAP depends on the quantification techniques and used to quantify threats for Commercial Off the Shelf Software systems T-MAP can tell to the management how cost effective the patching etc is. The attack path model is developed by using UML class diagrams. For each step, four class diagrams such as access class diagrams, vulnerability class diagrams, target resources class diagrams and affected value class diagrams are required. T-MAP has support of an automatic tool called Tiramisu. This tool depends on three types of input values: vulnerability, IT details of the institution and business impact of the institution on its IT assets. The tool then generates a detailed list of all the threats to the institution. The main functionary of this tool is data collection engine which gathers information about vulnerability automatically from a number of main sources such as NIST, Security Focus, CERT, CC and Microsoft. Then it converts this information into the form suitable for storage in the database. In the last this information is added into the data storage and management unit of this tool. CORAS based on UML is graphical threat modeling and specification language. By using CORAS undesirable behavior is documented in the form of threat scenarios. The CORAS profile offers specialized use case diagrams for modeling threats and unwanted behavior. CORAS also has a tool support called diagram editor. The CORAS threat modeling tool is just diagram tool to draw threat scenarios and has no analysis facilities. CORAS is based on Australian Risk Management Standard AS/NZS 4360:2004 and consists of the activities such as Establish the context, Identify risks, Analyze risks, Evaluate risks and Treat risks[8]. SSAI (Software Security Assessment Instrument) is a set of activities which use certain resources and tools to help in developing secure software applications. The first resource SSAI provides is an online vulnerability database containing information about various vulnerabilities and their exploits and mitigations. The second SSAI resource is a security checklist [9] that can be used to guide development process. The third resource is a list of publicly available static code scanning tools. SSAI also provides flexible modeling framework (FMF) which is a modeling tool. The FMF can be used to develop models and verify them for desired properties using model checking. Lastly, SSAI provides a property-based testing tool (PBT) which uses the security properties specified in the security checklist or FMF as a basis to test the software[9]. Correctness-by-Construction methodology integrates formal methods in critical phases of software development such as specification and design. This methodology helps to identify and eliminate flaws at the design level before testing and implementation. This methodology does not recommend the use of formal method in every phase of development but it recommends the application of formal methods only to security critical components of systems [12] and [2].

3. FORMAL METHODS

Formal methods are techniques predicated on mathematical logic and are utilized for the specification, analysis, design, testing and implementation of software systems[1]. By utilizing formal methods, applications can be developed in a precise way without ambiguities in a mathematical language. The resulting models can be analyzed, validated and verified by utilizing formal method tools such as theorem provers and model checkers[26]. In this way, we can check that the specification is consistent and rectify before implementation. Model checkers and Theorem provers can be utilized in conjunction or alternatively. The choice depends on the system requisites we are modeling. Formal methods avoid the desideratum to perform exhaustive testing of properties. By utilizing formal methods, applications can be analyzed for a gamut of values of data which is insurmountable by conventional testing techniques[25]. Formal methods can also be utilized for the validation of software systems. Since, application of formal methods requires an abundance of expertise, these methods has been utilized in limited areas of computer science such as critical systems. Some examples of formal methods are: Event-B, Z, VDM++, RODIN, Alloy, Z/EVES, Isabelle, Coq etc. Formal methods can be categorized broadly into three main categories: formal specification languages, model checkers and theorem provers. Formal specification languages such as Z can be used to for the specification of the requirements of the systems and to specify the design of the systems. Formal specification languages can also be used to generate test cases from the requirement document of the systems. Model checkers are used for the validation and verification of properties such as deadlocks. Theorem provers are used to generate formal proofs of the design of the systems and to prove properties such as syntax checking, type checking, domain checking, invariant preservation and proof obligations.

4. Security Properties

Security properties are the mitigation measures used to stop an attack if vulnerability is exploited and attack transpires. Vulnerability is an impuissant point in the system which can be compromised by an adversary in order to launch an attack on the system resources. A system resource is an entity which has a value in the system and its protection is compulsory for the secure operation of the system. A threat is a potential attack which may transpire. Security properties can be divided into six broad categories: Authentication, Authorization Confidentiality, Integrity, Availability and Non-repudiation. Authentication is the process in which the identification of a user or a process is established in the system. A valid user must have unique identification information already stored in the system. Authorization is the process during which access to system resources is determined for the users of the system. Confidentiality determines the read access to data. It determines that data is only available to those users who have rights to read it. Integrity determines the write access to data. It determines that data in the system can only be changed by felicitous persons in felicitous way. Availability means system resources are available to legitimate users when needed and perform acceptably. This means that users of the system can access the system resources at any time. The system must respond accurately. Non-repudiation means user cannot perform an action and later gainsay performing it. It transpires when a legitimate user performs an action and later on refuses to accept it. Therefore, a vigorous logging system is compulsory to control such a situation. There are many more security properties and mechanisms being used in industry but we will focus on the main security properties which are described above.

5. Basic Security Models

In this research, we basis our research hypothesis on the basic security models developed over the years by the experts in the area. The most commonly used security models are the STRIDE models and the DREAD model.

5.1 STRIDE Model

The STRIDE model covers the main six broad categories of threats that an application may face. STRIDE threat categories are Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. Spoofing is the process in which an illegal user gets the identification information of another legal user through wrong means and pretends to be a legal user. In this way user can gain access to system resources. In tampering, an illegal user modifies system data by gaining access to system. Repudiation is concerned with the refusal of an action performed by a legal user. It happens when logging systems is very weak or non-existent. In information disclosure private and personal data of users or institution is opened to other users who do not have rights to access that data. In denial of service the system resources are made unavailable to legitimate users. In elevation of privilege, a user who has access to certain resources gains access to other more privileged resources for which he or she is not authorized to access.

5.2 DREAD Model

DREAD model is used for rating risks of the threats generated by the STRIDE model. DREAD model covers the five broad categories: Damage Potential, Reproducibility, Exploitability, Affected Users and Discoverability. The values to these five parameters are assigned by the designer or user and then DREAD algorithm is applied to calculate the risk. Depending upon the results of the DREAD algorithm, threats are rated as High, Medium and Low. High means more danger and strong controls are required. Medium means moderated danger and medium level controls are enough. Low shows that threat is of low level and simple controls are enough. In some cases no controls are applied and threats are accepted.

METHODOLOGY

Formal Development Methodology for Secure Web Applications (FDMSWAP) consists of seven phases. The architecture of this methodology is shown in Fig.1. This methodology is based on the lightweight use formal methods. Lightweight means the use of formal methods in a subset of the activities of the methodology only, not on every phase. The basic theme behind this methodology is that use of formal methods should be applicable by software engineers for developing secure web applications. In FDMSWAP, Event-B and Z Notation have been used as specification languages for the specification of models. RODIN platform and Z/EVES theorem prover have been used for analysis, verification, validations and formal proofs of models. ProB model checker have been used for checking invariant preservation and deadlocks freeness in the system models. This methodology consists of 7 phases: requirements, abstract formal models of security properties, threat modeling, and system development, verification of secure models of the System, model checking and validation of secure models of the system.

6.1 Requirements

6.1.1 Requirement Specification of System (English)

In the requirement specification phase of this methodology, the system to be developed is analyzed. System entities are identified, system assumptions are highlighted, functional and non-functional requirements of the system are identified and written in English language. Also, required properties for validation are written in this phase.

6.1.2 Data Flow Diagram (DFD) of System.

An overall Data Flow Diagram (DFD) of the system is created. The DFD produced in this phase is used as an input for the SDL Threat modeling tool. The SDL Threat Modeling tool takes this DFD as input and produces a list of possible threats to the system.

6.2 Threat Modeling

6.2.1 Threats Modeling: Apply STRIDE Model

In this phase, STRIDE model described in 5.1.1 is applied to the DFD of the system produced in phase 6.1.2, to generate all possible threats to the system under consideration. This process is done automatically by using SDL Tool. In SDL Tool, STRIDE model is built in into the tool and is applied to each element of DFD of the system and then the tool automatically generates a list of all possible threats to the system.

6.2.2 Threat Rating: Apply DREAD Model

In this phase, DREAD model described 5.1.2 is applied to the threats generated in phase 6.3.1. The threats are rated according to their severity by using the threat rating algorithm described in 5.1.2. Threats are rated as high, medium and low

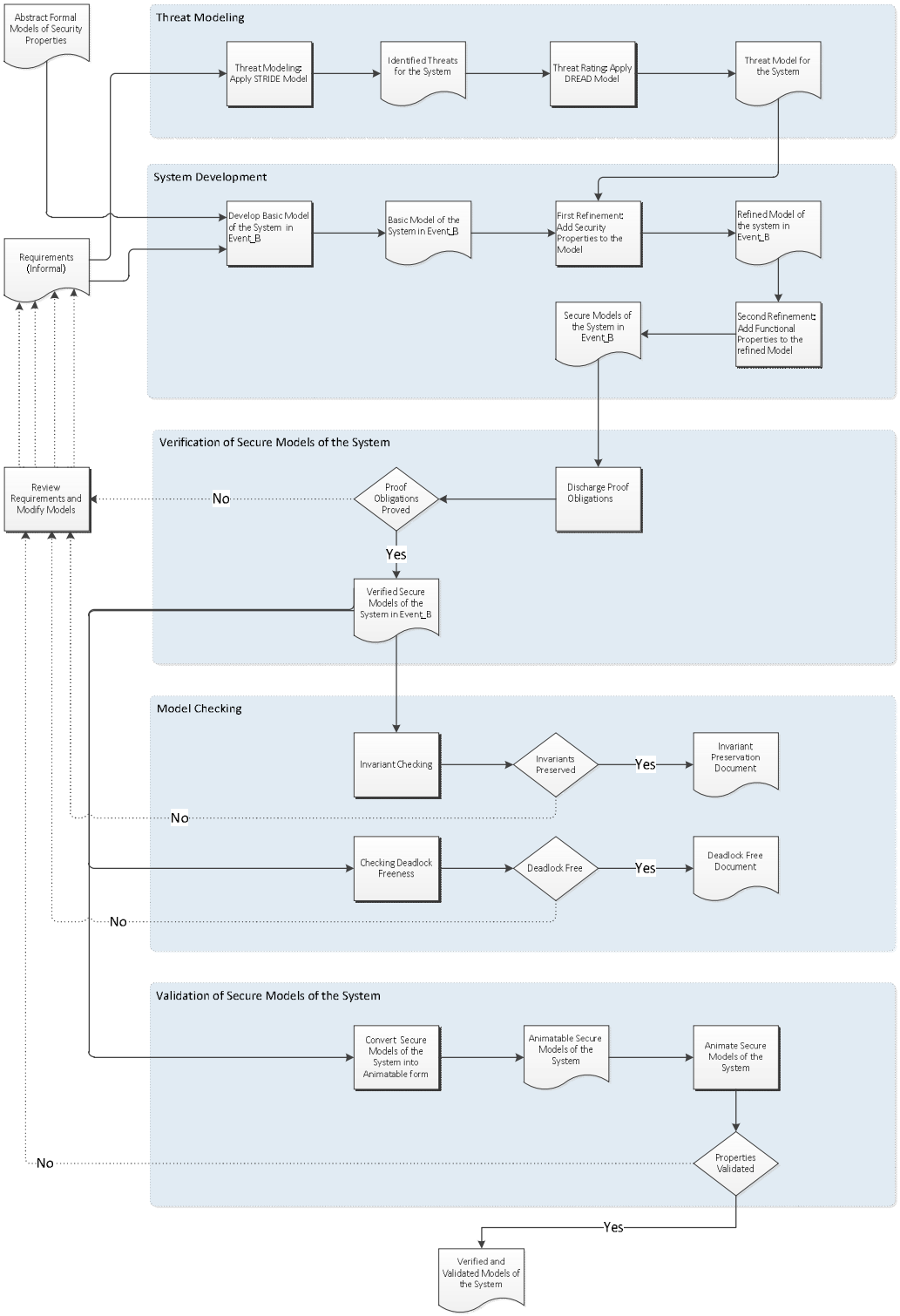

Fig.1Architecture of the Methodology

6.2.3 Threat Model of the System

In this phase, threats generated and rated in phases 6.3.1 and 6.3.2 respectively are analyzed. The threats which are based on the assumptions made in phase 6.1.1 are discarded. A threat model for the system consisting of this refined list of threats is obtained at the end of this phase.

6.3 Abstract Formal Models of Security Properties

Since this methodology is based on the use of formal methods for secure software development for web applications, therefore a formal model of security properties is developed at a very abstract level by using Z notation to get a higher level abstract model of security properties. The security properties are the mitigation measures used to stop an attack if vulnerability in the system is exploited and attach happens. Security properties used in this methodology are confidentiality, integrity, availability, authentication, authorization and non-repudiation.

6.4 System Development

6.4.1 Basic Models of the System in Event_B.

In this phase of this methodology, basic formal model of the system is produced in Event_B. Requirements described in 6.1.1 and abstract formal models of security properties are used as input to this phase. Basic functionality needed for the application of security properties and functional properties is added in this phase. This basic model consists of those functions performed by the administrator of the system.

6.4.2 First Refinement: Add Security Properties to the Basic Models of the System.

In this phase of this methodology, basic formal model of the system developed in phase 6.4.1 is refined to add more functionality. In this refinement, security properties based on the threat model produced in phase 6.3 are added to the basic formal model of the system. Purpose of refinement is to start modeling with simple operations and add more complexity gradually at each refinement.

6.4.3 Second Refinement: Add Functional Properties to the Refined Formal Models of the System.

In this phase of this methodology, the refined formal model of the system in phase 6.4.2 is further refined to add the functional properties of the system as described in phase 6.1.1. These functional properties define the behavior of the system.

6.4.4 Secure Formal Models of the System in Event_B.

In this phase, secure formal models of the system are produced in Event_B. These models are based on the refinements done in phase 6.4.2 and 6.4.3. These formal models have all the required security properties and functionality. For this methodology, only two refinements are required. But further refinements can be made if required. This may vary from system to system. For a simple web application with few functional requirements these two refinements are enough.

6.5 Verification of the Secure Models of the System.

In this phase of the methodology, secure formal models of the system produced in phase 6.4.4 are verified by theorem provers. In this methodology, Altelier B theorem provers, plugged into the RODIN platform have been used the verification of the secure formal models. Altelier B provers generate proof obligations about the system. These proof obligations must be discarded for the successful operation of the system. Altelier B provers are started by the user and they prove the proof obligations. Normally, majority of proof obligations are discarded automatically on the first run of the provers. If some proof obligations are not discarded then different tactics are applied to prove them including modification in the formal models of the system in consultations with the requirements.

6.6 Model Checking

In this phase of this methodology, the secure formal models developed in phase 6.4.4 are passed to ProB model checker, a plug in RODIN platform. I this methodology, only invariant preservation and deadlocks freeness are checked for the models of the system. If the system passes the both tests then system is acceptable otherwise modifications are made in the formal models of the system by using requirements of the system.

6.7 Validation of Secure Models of the System.

In this last phase of the methodology, properties written in the requirements phase 6.1.1 of the system are validated. These properties include security properties and functional properties. For validation AnimB animator is used. This animator is also a plug in for the RODIN platform. The constants and variables in the formal models of secure system are assigned values to make them animable by AnimB. Then AnimB is started and behavior of the system is observed. If behavior of the system is coherent with the properties defined in the requirement phase then the models are considered as validated otherwise modifications are made in the formal models of the system by consulting requirements of the system.

7. Conclusion.

In this paper, a new methodology to develop secure web applications has been presented. This methodology is based on the lightweight use of formal methods. A brief survey of relevant literature has been presented. An introduction to formal methods is presented in the paper. Formal methods are tools and techniques based on mathematical logic and can be used for the specification, verification and validation of software systems. Security properties are the key for any successful software system. These are briefly presented in this paper. These security properties include authentication, authorization, confidentiality, non-repudiation, denial of service and elevation of privilege. An introduction to the basic security models STRIDE and DREAD is also presented in this research. In methodology, there are 16 phases starting from requirements to ready to code models. In the future, the methodology will be applied to a real world case study in which a simple web based application will be developed and methodology will be validated. We will also investigate for automatic conversion of formal models into code in the future.

REFERENCES

1. Ali, G., Khan, S.A., Ahmad, F., Zafar, N.A., 2012. Visualized and Abstract Formal Modelingtowards the Multi-Agent Systems, *J. Basic. Appl. Sci. Res* 2(8), 8272-8284.

2. Amey, P., 2002. Correctness by Construction: Better can also be Cheaper. *CrossTalk: the Journal of Defense Software Engineering*, 2, pp.24–28.

3. Apvrille, A. &Pourzandi, M., 2005. Secure software development by example. *Security & Privacy, IEEE*, 3(4), pp.10–17.

4. Braber, F. et al., 2007. Model-based security analysis in seven steps—a guided tour to the CORAS method. *BT Technology Journal*, 25(1), pp.101–117.

5. Chen, Y. & Boehm, B.W., 2007. Stakeholder value driven threat modeling for off the shelf based systems. In *Software Engineering-Companion, 2007. ICSE 2007 Companion. 29th International Conference on*. pp. 91–92.

6. Chess, B. &Arkin, B., 2011. Software security in practice.*Security& Privacy, IEEE*, 9(2), pp.89–92.

7. De Win, B. et al., 2009. On the secure softwaredevelopment process: CLASP, SDL and Touchpoints compared. *Information and software technology*, 51(7), pp.1152–1171.

8. Fredriksen, R., Kristiansen, M., Gran, B., Stølen, K., Opperud, T., Dimitrakos, T., 2002. The CORAS framework for a model-based risk management process. Computer Safety, Reliability and Security 39–53.

9. Gilaninia, S., Mousavian, S.J., Taheri, O., Nikzad, H., Mousavi, H., ZadbagherSeighalani, F., 2012. Information Security Management on performance of Information Systems Management,*J. Basic. Appl. Sci. Res* 2, 2582–2588.

10. Gilliam, D.P. et al., 2001. Development of a software security assessment instrument to reduce software security risk. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001. WET ICE 2001. Proceedings. Tenth IEEE International Workshops on*. pp. 144–149.

11. Guan, H. et al., 2011. Environment-driven threatselicitationforweb applications. *Agent and Multi-Agent Systems: Technologies and Applications*, pp.291–300.

12. Guan, H. et al., 2011. STRIDE–Based Risk Assessment for Web Application. *Applied Mechanics and Materials*, 58, pp.1323–1328.

13. Hall, A. & Chapman, R., 2002. Correctness by construction: Developing a commercial secure system. *Software, IEEE*, 19(1), pp.18–25.

14. Hussain, S., Erwin, H. & Dunne, P., 2011. Threat modeling using formal methods: A new approach to develop secure web applications. In *Emerging Technologies (ICET), 2011 7th International Conference on*. pp.1–5.

15. Ingalsbe, J.A. et al., 2008. Threat modeling: diving into the deep end. *Software, IEEE*, 25(1), pp.28–34.

16. Jiang, L. et al., 2011. A Security Evaluation Method Based on Threat Classification for Web Service. *Journal of Software*, 6(4), pp.595–603.

17. Kumar, M.U. et al., 2010. Agile Modeling for Security Requirements–Embedded Application Case Study. *International Journal of Engineering Science*, 2(6), pp.2014-2019.

18. Muniraman, C. &Damodaran, M., 2007. A practical approach to include security in software development. *Issues in Information Systems*, 2(2), pp.193–199.

19. Nikakhtar, N., Jianzheng, Y., 2012. Risk Management Model of Electronic Commerce by the help of Decision support system, *J. Basic. Appl. Sci. Res* 2(1) 118-124.

20. Potter, B., 2009. Microsoft SDL Threat Modelling Tool. *Network Security*, 2009(1), pp.15–18.

21. Saini, V., Duan, Q. &Paruchuri, V., 2008. Threat modeling using attack trees. *Journal of Computing Sciences in Colleges*, 23(4), pp.124–131.

22. Singh, A.A. & Singh, K.S., 2012. Network ThreatRatings in Conventional DREAD Model Using Fuzzy Logic. *International Journal of Computer Issues*, 9(1), pp.1694–0814.

23. Singhal, A., 2011. Development of Agile SecurityFramework Using a Hybrid Technique for Requirements Elicitation. *Advances in Computing, Communication and Control*, pp.178–188.

24. Sodiya, A.S., Onashoga, S.A. &Oladunjoye, B.A., 2007. Threat modeling using fuzzy logic paradigm. *Informing Science: International Journal of an Emerging Transdiscipline*, 4(1), pp.53–61.

25. Steven, J., 2010. Threat Modeling Perhaps It's Time. *Security & Privacy, IEEE*, 8(3), pp.83–86.

26. Zafar, N.A. &Alhumaidan, F., 2011. Transformation of Class Diagrams into Formal Specification. *IJCSNS*, 11(5), pp.289-295.

27. Zafar, N.A., 2009. Formal specification and validation of railway network components using Z notation. *Software, IET*, 3(4), pp.312–320.