Threat Modeling in Web Applications

Soumya Ranjan Satapathy

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
June 2014

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Information Security)

by

Soumya Ranjan Satapathy

(Roll No.- 211CS2368)

under the supervision of

Prof. Durga Prasad Mohapatra

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Threat Modeling in Web Applications*** by ***Soumya Ranjan Satapathy*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology with the specialization of Information Security in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: May 30, 2014

(Prof. Durga Prasad Mohapatra)
Professor, CSE Department
NIT Rourkela, Odisha

Author's Declaration

I hereby declare that all work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been submitted for any academic degree. All sources of quoted information has been acknowledged by means of appropriate reference.

Soumya Ranjan Satapathy
Roll: 212CS2368
Department of Computer Science

Acknowledgment

I would be starting with thanking **the Almighty** for his gracious kindness during the tenure of preparing the thesis and complete the same. I can only thank him, no more is under my ability.

I am indebted to many local and global peers who have contributed towards the execution of this thesis. At the outset, I would like to express my sincere gratitude to **Prof. Durga Pr. Mohapatra** for his all around support during my thesis work right from its inception. Being my Supervisor, He has always motivated and supported me to stay focused on achieving my goal. His insightful advices, anytime help and observations have greatly helped me in gaining the overall willpower in me and to move forward with research work in depth. He has been a sheer source of knowledge and it has helped me whenever i needed any.

I am very much indebted to **Prof. Santanu Ku. Rath**, Head of the department, CSE, for his continuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors, especially **Prof. B. Majhi** and **Prof. S. Mohanty** of my department for their support.

I would like to thank **Tata Consultancy Services, Bhubaneswar** for choosing me for its Internship program for 12 months. It gave me a platform where I could really see and have the practical implementation of my knowledge. I do not think it would have been the same if the Internship had not happened to me.

I must acknowledge the academic resources that I have got from **NIT Rourkela**. I would like to thank administrative and technical staff members of the Department who have been kind enough to advise and help in their respective roles.

Last, but not the least, I would like to dedicate this thesis to **my family**, for their love, patience, and understanding.

Soumya Ranjan Satapathy

Roll-212cs2368

Abstract

Todays competitive and profit-driven online environment needs a web application to be much secure as it is going to be tested in all possible ways by the attackers for any sign of vulnerability which can be converted into a big success for him to gain control to the maximum of the software. In order to produce a secure application, it has to be securely built right from the design phase throughout the software development life cycle. The most effective methodology of implementing this is threat modeling. There have been a lot of improvements and researches on the process of threat modeling and its approaches. Following these, Some tools are developed by some Enterprises to support the process of systematic threat modeling.

In this thesis, the most widely accepted process of threat modeling, that has been proposed by Microsoft, is explained along with other approaches for it. Two industrial projects, with the support of Microsoft SDL tool for Threat modeling have been threat modeled and discussed. Towards the end, some modifications to the hybrid approach of threat modeling have been proposed and have been implemented on the open source workbench supporting that approach.

Key words: Threat modeling, security in web application, hybrid threat modeling approach, STRIDE, DREAD, Security Development Life Cycle(SDLC)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's hostile and competitive Internet era, a web application is very much likely to be assessed thoroughly from all possible ways for its inherent vulnerabilities that can be exploited by an attacker. As the proverb goes "thieves are more intelligent than cops", even a least sign of weakness can be converted to a big disappointment for the software system by the high intellectuality of the attacker. As a consequence, the data gets revealed that has to be kept secret, the system gets compromised, unable to serve or crashed, reputations and trust of organization at stake and many more miserable consequences. So vulnerabilities have to be minimized. Software API, datastore, data transfer channel etc. are the most important lines of defense for protecting critical information assets in utility applications like e-commerce, e-banking, e-forecasting systems where there is a large amount of confidential data processing involved. Vulnerabilities in a software application is beyond the capabilities of the OS or Network level security mechanisms or intrusion detection techniques as they lack the knowledge of application semantics as discovered by D Xu and KE Nygard in [1](semantics means particular nature of individual application in terms of its control and data flow). Reliance on network security alone or installation of firewall is not sufficient as it does not address the logic errors, flaws in architecture of software system, flaws in operating system and its resource limitations or the design level problems. As it started, On 2^{nd} Nov 1988, an Internet worm in the UNIX operating system was created by a 22-year old student named Robert Morris which was capable of exploiting vulnerabilities by using buffer overflow attacks. In those days, instal-

lation of firewall with an proper application proxy was considered to be sufficient for security. But this worm contradicted this fact and posed a challenge for the security designers. From that day till today there have been inventions of a lot of attacks that are gradually becoming more sophisticated requiring less intruder knowledge. The fact that Web-based malware attacks doubled in the second half of 2013 in comparison with the first half, according to the latest threat report from F-Secure Labs [2] suggests the explosive growth of threat portfolio against the web based applications. So on the basis of the last two or three decade's security trend, innovative threat evaluation techniques for computer systems and software systems are required. From the business point of view, the security objectives should address the areas like identity management, financial risk, business continuation, corporate reputation along with legal and regulatory perspectives properly. Risk management is a major goal in business applications, ie security resources are applied to vulnerabilities that pose great risk to the business.

In the year 1968, there was a conference organized by NATO science committee on software engineering where the main discussion was on software crisis and how they can be addressed by software engineering principles. This goal gradually gave birth the fine-tuned field of software engineering in which the formal step by step practices are being used today were evolved (broadly the steps are: requirement analysis, software design, implementation, software testing, software deployment and maintenance). Now-a-days the growth of internet and telecommunication has given rise to the new type of crisis: software security crisis, which is the result of casual security considerations and negotiations over it. To address such a crisis, secure software engineering is needed and the process of Security developement life cycle to be considered along side of Software developement life cycle.

In a software developement life cycle(SDLC), for a long time, security has been considered as a non-functional requirement. Functional requirement is defined as the system of requirements which depicts the fuctions that the software system is desired to do. So this type of requirement defines the behavior of the software system. On the other hand, non-functional requirement is the system of require-

ments which includes all other aspects than the functional ones like assessment of cost, platform compatibility, monetary and profit oriented decisions etc. So adding security requirements to the non-functional class is fine as security requirements do not come under what a software system is required to do. Rather, security requirements define the behavior that the system should have when an undefined or unknown function or situation arises. Being taken as a non-functional requirement, security had been taken as an after-thought or of lesser priority. It was not compulsory to make softwares security aware. But today the scenario has been changed. Now large amount of user data reside in the application database and in process memory during execution of the operation. Hence a secure measure over every operation is essential. Hence security measures can be taken as 'inherent' in the requirement given by the customer. Now-a-days security cannot be treated as a after-thought as a little bit security flaw leaves room for big exploitation that can be performed by the attacker. Security requirements and functional requirements have to go side by side. Security implementations and functional requirement implementations have to be done side by side and interdependently. Hence it is not at all arguable if the security requirement are considered as functional requirements.

In fact there are several benefits if security is considered as functional requirement and it's considered in the software development life cycle starting with the requirement analysis phase. firstly, focus on security aspects and a more detailed view along with its cause and effects on the performance and functionality can be analyzed and obtained which helps the designer find out the countermeasure against each threat right from the earlier stage of SDLC. The detailed discussion has been done by G. Sindre and A. L. Opdahl in [3]. Secondly, the developer can be well aware of the security aspects of the software before developing it. Hence the security testing cost in turn gets reduced. These two advantages make the software become secured right from its inception and on successful completion a secure system comes out which is secured against too many types of attacks.

When talked about security in a software or web application as it has been talked about in the previous paragraphs, it essentially means the existence of three

aspects: confidentiality, integrity and availability. In a broad sense, confidentiality is the process of preventing unauthorized disclosure, integrity is the process of preventing unauthorized changes and availability is the prevention of unauthorized access. Information security means to protect information from unauthorized access, disclosure or change. Information security includes another aspect in case of a software system: availability and recovery of the responsibility of information keeping the information system or software system running fine while the system performs its designated functionalities and protecting the resources from any unnecessary and unintended situations.

Security in a web application can be incorporated at the design phase or after deployment ie the maintenance phase. Incorporation in the design phase is the extensive practice of understanding the system assets that are to be protected, deployment environment, data flows and control flows, types and number of users to access the system once deployed, available resources that are going to be utilized during the operations of the software, all possible cases of misuse that can happen over each resources or processes and many more. The system designer produces the design document keeping all these aspects in mind and the next phase ie implementation phase starts. The failure to produce secure design document and the respective secure code would eventually lead to exploitation of the possible vulnerabilities by the attacker. Also, relative cost and negative effects on security return-on-investment(ROI) to fix these vulnerabilities proves to be highest after deployment, which is calculated to be 30 times as high as the cost to correct the faults at the design phase as described by Microsoft Corporation and iSEC Partners in [4]. Hence the second option of incorporation of security in the maintenance phase has been proved to be costly. In contrast, during the testing phase, there is a security testing conducted to test for all possible kinds of threats. The success of this testing depends on the robustness of the design phase security incorporation. Hence at the design phase the security aspects are best added which is implemented in the implementation phase. Next comes the cost effectiveness consideration which has also to be done in the design phase in which the only

necessary security implementations in the software is done leaving the not-much-needed parts that may unnecessarily consume cost and time. Threat modeling is the process of design level security consideration(consideration includes identification, prioritization and mitigation) and to cost-effectively do it, risk-based threat modeling is considered. Before the introduction of threat modeling, the exact meaning of threats, vulnerabilities, exploitations, attacks and difference between them should be understood.

1.1 Basic terminologies

Threat: A threat is something danger that may disrupt the operation, working procedure, integrity, or availability of a software or a network. This can take any form and can be malevolent, accidental, or simply an act of nature. In other words, threat is a possible danger that might exploit a vulnerability to breach security and thus cause possible harm.

*Vulnerability:*It can be defined as an inherent weakness in the design, configuration, implementation, or management of a network or system that renders it susceptible to a threat. Vulnerabilities are what make networks susceptible to information loss and downtime. Every network and system has some kind of vulnerability.

*Exploitation:*An exploit is the way or tool by which an attacker uses a vulnerability to cause damage to the target system. The exploit could be a package of code which creates packets that overflow a buffer in software running on the target, which is also known as buffer overflows. Alternatively, the exploit could be a social engineering scheme whereby the bad guy talks a user, preferably an employee into revealing sensitive information, such as a password, over the phone.

Attack: An attack is any attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an asset.

1.2 Threat modeling

Threat modeling is a procedure for optimizing network security by identifying objectives and vulnerabilities, and then defining countermeasures to prevent or mitigate the effects of threats to the system. It has emerged as an independent and comprehensive methodology. Many researches have been taken place for the advancement of this area. Threat modeling assures security to a higher level of abstraction. By understanding the threat scenarios of the system and the appropriate mitigation plans available, it helps to find out exact vulnerabilities to particular assets, serves to produce secure design, so thence secure implementation framework and at last penetration testing in the context of the application security life cycle as described in books of Frank Swiderski and Window Snyder [5] [6]. Hence again it can be defined as a structured and formal approach of presenting, assessing and documenting security risks of a particular software.Threat modeling cannot explicitly be considered as mathematical science and hence wit the freedom and flexibility even a non-security expert can exercise it with a provided convenient framework and support(though not with full efficiency).

As discussed before, its better to add security suggestions right in or before the design phase of the software development life cycle. The same is followed by threat modeling mechanism. It is documented by the designer with proper knowledge of system requirements, deployment environment, system environments, security requirements and the resources available for the system. Taking all into consideration, the model is documented. Microsoft states, Starting the process of threat modeling early in the SDLC is important since it haves the capability to reveal the weakness in architecture that may require significant modifications to the product. Peter Torr in [7] explains that making early design level modification is more cost-effective compared to it done at some place later.

1.3 Different approaches of threat modeling

1.3.1 Asset-Centric

Asset-centric threat modeling involves starts with identifying critical assets. Assets are the interfaces that are entrusted to a system, such as a collection of sensitive personal information. It involves assessing the risks associated with them, approximating them and ranking the risks.

1.3.2 Attacker-Centric

Attacker-centric threat modeling starts with the attacker objectives, motivation and capabilities. Objective means this evaluates their goals, and how they might achieve them. Attacker's motivations are often considered and given importance than any other factor. Capabilities are the level of harm that can be done and the entry points where it can be done and hence involves identifying points, evaluating attack path, evaluating damage potential and risk rating. This approach usually starts from either entry points or assets. This approach usually starts from either entry points or assets. Generally, Attack tree is used to describe the attacker-centric approach. Attack tree representation has been stated in detail in the literature review section(Chapter 3).

1.3.3 Software-Centric

Software-centric threat modeling, also termed as 'design-centric,'system-centric', or 'architecture-centric', starts with the design of the system. It involves application decomposition and profiling, identifying threats for scenarios and mitigation strategies. It attempts to step through the model of a system, looking for types of attacks against each element of the model. The design-centric threat modeling may start with Data Flow Diagrams(DFD) or Unified Modeling Language(UML) diagram. In other words, this type of modeling may use data flow scenarios or control flow scenarios as its input on which threats are to be assessed. This is elaborately explained in literature review section(chapter 3).

All the three approaches have their own significances. Each approach is taken

at particular time according to the requirements of the system. We can use hybrid approach also for better results.

1.4 Risk-Based Threat modeling

The application being developed by an organization is always time-bound and constrained by the assigned budget. The cost estimated for the whole software development life cycle gets followed by the developing teams(Schedule slippage is highly discouraged). The design phase and so the threat modeling is also assigned some time limit which contains excessive amount of discussions, meetings and amendments in documentations. Hence it's costly to resolve each and every threat that comes along the way since its highly time and labour consuming. Again, the threats that are not likely to be exploited as vulnerabilities(ie those assets which can't be attacked or where there is no entry point for an attacker) need not be taken care of which will ultimately save time. The ones which will be high risky assets and need to be mitigated and which don't need so much of it is decided by risk analysis. Prominently, DREAD methodology, developed by Microsoft(OWASP 2001), is the process of analyzing threats inside the assets of a system. Details of DREAD methodology is described in the literature review section (chapter 2). As a result, an abstraction of threat modeling is done based on priorities of threats by which the higher priority threats are resolved in the first run and subsequently the lower priority threats.

1.5 Motivation

The threat modeling done upon the hybrid approach uses the concept of Misuse case diagram to identify the threats and attack trees to represent the threat. There is no scope for report generation. In industrial projects, a design document, represented as a report, works as a workbench which is referenced throughout the rest of the phases in SDLC. Again from the paper [8] published by Advanced Strategies, Inc., it is better to use DFD instead of using UML diagrams for modeling of applications since it is more important to model the information flow rather than showing functionalities of system in UML(which is told to be a misuse of UML diagram in paper [8]). Hence its better to use the hybrid approach with the use of DFD instead of UML with the report generation capabilities.

1.6 Problem Definition

The objective is to survey and explain the threat modeling process followed at the software industry today and using the same approach, perform threat modeling for some live web application systems. In the next part, to explain the existing hybrid approach [9] proposed by Asoke K Talukder et al for threat modeling and introduce some modifications to it, which is done by using data flow (DFD) for the information flow modeling purpose instead of the misuse case diagram used that models the functional behavior, applying STRIDE to the DFDs and finally by generating the threat reports based on the user input and mitigation suggestions.

1.7 Organization of the thesis

A complete explanation of threat modeling along with the proper mitigation techniques for different types of attacks has been stated in this thesis. The threat modeling tasks have been carried out for the live industrial applications being developed at Tata Consultancy Services, a leading India originated Information Technology organization(arguably, the most successful one in India). A hybrid approach has been stated and a modified implementation of it for better usefulness of it has been stated towards the last part of the thesis.

The rest of the thesis is organized as follows.

In **Chapter-2**, an outline of security development life cycle (SDLC) followed in industries has been described with the detailed explanation of a case study of the SDLC process of Microsoft.

In **Chapter-3**, a detailed idea of the threat modeling process, risk analysis, concepts used in it, its approaches followed by the mitigation plans have been clearly described. In the second section of this chapter, the hybrid approach to the threat modeling has been explained in detail.

In **Chapter-4**, The threat modeling in two business applications has been explained doing one in greater details along with the mitigation plans.

In **Chapter-5**, the modification to the hybrid approach has been described along with implementation on the existing tool supporting the hybrid approach.

In **Chapter-6**, The conclusion has been stated followed by future scope.

Chapter 2

Security Development Life Cycle

2.1 Introduction

It can be defined as a software development process schedule which makes us build more secure software and can address to the security compliance requirements with the achievement of development cost reduction. Software-centric Threat modeling, that has been discussed previously, is synonymous to security development life cycle. The proper security development life cycle (SDLC) was described by Lipner and Steve in the paper [10] in which the detailed process of Microsoft SDL has been explained.

Microsoft has developed its own security development life cycle which is described in the paper by Lipner and Stevewith [11]. The aim of reducing software maintenance costs and increased reliability of software concerning software security related bugs etc are circumscribed into the Security development life cycle. Microsoft also descibes its own approach in [12]. The IT industries are not uniform. So individual companies use their own interest of SDLC according to the suitability of human talent, organizational size, security requirements, resources available (time, talent, and budgets) and many other. Success or failure of an application often relies on these dependent factors. The effect of these intangibles can be controlled by going through the basic blocks of good security development practices and understanding the implementation priorities based on the experience and maturity level of the development team. Though many researches are continuously and rigorously performed in order to achieve significant amendments, the

approach of Microsoft and its updates are more or less widely accepted by many IT companies for secure design purpose.

The overview of the above mentioned approach has been stated in the next section.

2.2 Microsoft SDL Optimization model

The Microsoft SDLC is based on three core concepts: education, continuous process improvement, and accountability. The investments on continuous education , huge practical dataset collection and training for job roles within a software development helps organizations to face adequately to the changes in technology and the dynamic nature of threats. The SDL gives heavy importance on understanding the cause and effect of security vulnerabilities in applications and begs regular assessments and amendments towards betterments of SDL process keeping in view the non-static nature of threats, the modernization of technologies and advancements in threat technologies. The collected Data is utilized to evaluate effectiveness of training, in-process metrics are being used to evaluate process compliance and post-release metrics assist in future changes.

The SDLC is represented in the sequence same as the phases of the traditional software development life cycle (SDLC). But here the additional activities performed in order to add some degree of security benefits over the conventional one. The SDL Optimization Model is divide into five phases roughly :

1. Training, policy, and organizational capabilities
2. Requirements and design
3. Implementation
4. Verification
5. Release and response

The first stage may be excluded from the stage schedule if it is considered to be a one time activity. It happens when the same type of software is being developed again and again with no need of extra knowledge and training. In that case it can be treated as a pre-SDL activity.

Figure 2.1: SDL Optimization Model with capability and maturity levels

Additionally, the SDL Optimization Model defines four levels of maturity for the capabilities and practices in these above mentioned phases. They are: Basic, Standardized, Advanced and Dynamic.

The Microsoft SDL Optimization Model starts with the 'Basic' level of maturity where there is little or no process, training, and tooling in place, and goes step by step towards the Dynamic level, which depicts the complete SDL compliance across a complete application. A sophisticated security application is generally built in (or expected to be built in) advanced or dynamic level of maturity. Again, Focus has to be drawn on the accuracy of the outcome after each stage. Each stage descriptions along with the guidelines for a proper execution of the schedule is described below.

2.2.1 Training, policy, and organizational capabilities

All the resources of a development team should get well informed and trained according to the specific security requirements of the software, the security basics, the on-going trends in security and management of privacy. This training can be continued on a scheduled manner in a year which the technical persons(design persons, developers, testers etc.) are mandatory to attend.

The training areas include

1. Secured design: It is concerned with topics like reduction of attack sur-

face area, basic understanding of defense, least privilege adherence, the security adopted by default etc.

2. Threat modeling: It includes topics like threat modeling overview, designing of a threa model, implementation constraints sticking to the threat model etc.

3. Secure coding: It includes understanding of buffer overflow(in C, C++), arithmatic errors(in C, C++), XSS, SQL Injection, weak cryptography etc

4. Security testing: it includes understanding of the difference between functional testing and security testing, risk assessment, methods of security testing etc

5. Privacy: it is concerned with topics like privacy sensitive data types, best practices of privacy design, assessment of risk, best practices of privacy development, best practices of privacy testing etc.

6. miscellaneous: topics like advance security architecture and design, dependable UI design, detailed studies of security flaws and vulnerabilities, implementation of manual threat mitigation etc.

2.2.2 Requirements and design

Security requirement

For a secure software system development, security and privacy need to be considered side by side. So the most crucial time to include trustworthiness to the application is the design phase. The early functional requirements by the customer lets the organization identify important milestones, deliverables, permissions along with the privacy and security aspects of the system (that might be explicit or implicit to the system).

Quality Gates/Bug Bars

There is use of quality gates and bug bars for the establishment of minimum acceptable level of security and the extend of privacy. These are certain threshold values of risks and severity respectively defined by the proper understanding of associated risks. Bug bar is set once only and cannot be changed any more. A development team negotiates the quality gates for each development phase.

Team should get them approved by the security personnels who might manipulate project-specific clarifications and more appropriate security requirements.With all the clarifications and improvements, Final Security Review (FSR) is completed.

Security and Privacy Risk Assessment

Security and privacy risk assessments (SRA and PRA) are processes that identify the functional faces of the application demanding deep review. The informations in such assessments are like find out the modules of the project that need threat models beforehand of release, modules of project demanding security design reviews before release, portions of the product that need penetration testing by a mutually agreed upon team external to the developing team, if any other testing or analysis required from the security point of view, the specific scopes of fuzz testing requirements, the privacy impact ratings(whether high privacy risks , moderate privacy risks or low privacy risks) etc.

design requirements

The development team should understand the difference between secure features and security features. Secure features are the features whose functionalities are well engineered in accordance to security, including extensive validation or cryptographic implementations of the data. Security features can be defined as the program functionality with security implementations(eg. firewall, IPSec, kerberos or SSl etc). So there is a chance that implementation of security features are added but still the system is left as insecure. The difference has to be well understood. If the security feature is the cause, the secure feature is the effect. The security design requirement includes the required actions that may include the security and privacy design specifications, specification review and/or the minimum requirement of cryptographic specifications. A good design specifications describes the complete and accurate secure implementation of all functionality provided by a given feature or function or in other words secure deployment information in a function.

Attack surface reduction

It means giving the attacker the minimum scope to attack on the system there by reducing the attack surface and vulnerability. It includes the roles of least privileges and limited access to users, implementation of layer defense etc to hide the exploitable spot from the attacker.

Threat modeling

This is what the whole thesis is about. It allows development teams to analyze, document and mitigation suggestion of the potential threats in design level models on an abstraction of risks associated. The documentation as the output is adhered to throughout the rest of the phases for a secure product development.

2.2.3 Implementation

Use of approved and updated tools

The developing organization should publish the approved tools along with their associated security checks, such as compiler/linker options and warnings, endorsed by the security adviser. The development teams should use the latest version of the developing tools to which out dates the previous security flaws and errors.

don't use Unsafe Functions

The existing functions, commonly used functions are always under scan in the attacker's eye for some vulnerability. So the APIs, common functions should be analyzed properly in the current threat environment by the security advisors before using them. All the prohibited or black-listed functions should be avoided from use by the developing team.

Static analysis

The source code should be put to Static analysis as it provides the scalable capability for performing security code review and also helps to confirm whether the secure coding policies are being followed or not.

2.2.4 Verification

Dynamic program analysis

It is the verification of the system at run time. It is required to confirm whether the program works as the design document demands. This task includes verifications of user privilege issues, memory corruption, and other critical security problems. Generally tools are used for the verification purposes for accuracy and automation(example of a tool: appverifier)

fuzz testing: Its a run time analysis by introducing random flaws to the application and check for the response.

Threat Model and Attack Surface Review

This review tracks any design or implementation changes to the system other than the design specifications and any new attack vectors being introduced because of the changes. These attacks are mitigated after detailed verification.

2.2.5 Release

Incident Response Plan

In worst, it might be the case that programs with absolutely no known vulnerabilities at the time of release may also be subject to new threats that may be discovered in future. For staying safe against such situations in future, an incident response plan is prepared. This includes an identified sustained engineering (SE) team to work in a security need after release which should be available 24*7 even on phone calls.

Final Security Review

Prior to release, it is the detailed assessment of all the security activities perormed in a application system by the security advisor with the assistance from the technical development personnels and the security and privacy team personnels. The FSR generally includes an assessment of the threat models, tool output, input and output validations, exception requests, performance issues against the previously standardized quality gates or bug bars et cetra. A FSR may be considered to be

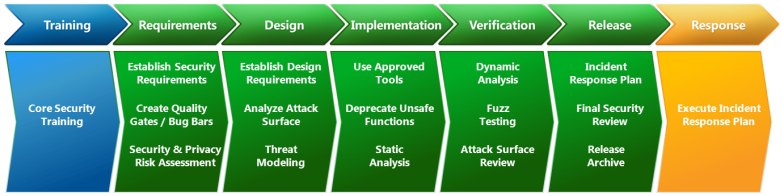

Figure 2.2: The Microsoft Security Development Lifecycle - Simplified

passed if the issues are fixed and mitigated properly, Passed FSR with exceptions if all the security and privacy issues identified by it are fixed/mitigated or all exceptions are satisfactorily resolved and FSR with escalation if the product does not reach to an acceptable compromise in terms of security.

Besides these 5 stages, there are some other security tasks that are carried out which may be

manual code review: Performed by highly experienced and skilled security persons focused around the critical assets that are utmost susceptible to vulnerabilities.

Penetration testing: It's a white box security analysis of an application system performed by the experienced security professionals which simulates the action of an attacker. It s objective is to discover the potential vulnerabilities present in the system because of failure in secure coding, fault in deployment environment etc. It is a very useful technique.

Vulnerability Analysis of Similar Applications: The vulnerabilities found in a similar software can also be present in the current application which may be left untouched by all the previous activities. Many information is searched over Internet and the vulnerabilities are tried to be uncovered with maximum effort.

Chapter 3

Related Work

3.1 Threat Modeling Overview

As it has already been discussed, threat modeling is a structured process of identifying and documenting the vulnerabilities to threats with a proper risk analysis associated with a system. It also can be treated as a security review in design review technique. As a matter of fact, the designers and the technical persons should understand the difference between secure and insecure system. A system generally does what it should do, but a secure system focuses on the fact that the system does not do what it should not do.

Threat modeling is a too complicated task if the application is considered as a whole, rather it gets simplified when it is done for specific components of the system and at last they are combined as a whole. So for doing this,the application need to be decomposed to small modules, all the dependencies are found out and the interfaces which can also be called as entry points are found out for the users and databases. Then the threat modeling process continues.

3.1.1 The process of Threat modeling

The process of threat modeling starts from defining the trust levels to each entry point. Trust level defines the level of the entry point up to which it may be dependable for interaction of data. There are mainly three types of trust levels though more may be obtained for complex applications namely administrator, user and un-trusted. The administrator trust level is concerned with the admin

module which carries full access to the system there by taken to be the most trusted one. The user trust level defines the interface with the user to the system which may subject to different types of attacks since the user is not always dependable, showing moderate level of trust. The un-trusted one, as the name suggests, is the most exploitable one to threats and mostly it is open to anonymous users to operate on. It demands careful security concern and resource managements. The process of threat modeling have been proposed in many papers. Steven F Burns in [13] descibed the importance and benefit of considering application security at design phase as well as defined process of threat modeling which used data flow diagram to serve the purpose. Scott ambler described the complete process of threat modeling in [14]. John Steven in [15] analyzed the importance of threat modeling in web applications in current scenarios. In [16],Danny Dhillon showed the real world experiences and challenges faced while developing the threat model in EMC corporation, by using Microsoft's approach. . The process of threat modeling has been explained and illustrated on online banking system in paper [17] by C Mockel and Ali E. Abdallah, and by Ebenezer A Oladimeji et al in [18]. A case study on an e-learning system was taken for threat modeling by Maria Nickolova and E. Nickolov and described in [19]. A complete system of OpenFlow, a network application was threat modeled by Rowan K. using Data flow diagram and described in [20]. Suvda Myagmar, A J Lee and W. Yurcik applied Threat modeling to three systems: Software-Defined Radio, a network traffic monitoring tool (VisFlowConnect), and a cluster security monitoring tool (NVisionCC) was applied and described the process in [21]. Though many academic as well as industrial organizations have undergone many researches on the process of threat modeling, the most accepted one has been that of Microsoft, which encompasses all the aspects of security and taking all together it brings out a documentation that guides through the rest of the process. P. Torr in [7], S.Hernan et al in [22] have described the practical approach to the STRIDE based threat modeling approach which is proposed by Microsoft in [23].

The detailed process of threat modeling has been described in the following

section is proposed first by Microsoft, which has been getting followed by many information technology organizations.This has been most successful approach to be followed by most practical applications and has been followed by all the above mentioned papers. The process has been depicted by taking a trivial example of Student Grade Display system for more understanding purpose.

Before going into the complete details of the threat modeling process, a brief introduction should be given of a data flow diagram that is going to be used in the approach of threat modeling process proposed by Microsoft.

Data Flow Diagram :

This is the diagram that is used in the requirement analysis as well as in the design phase of the software development life cycle. It is a pictorial representation of the flow of data in the system, modeled from the process aspect. It is regarded as the visualization of data processing. The data flow diagram depicts the interactions of the system with other systems and external objects in terms of data along with the interaction of data within the system. Being simple, it has its own short coming: it does not have the capability to depict timing information and parallel processing. The DFD is related to structural programming as well as it can be linked to the object model. (in contrast, UML is related to object-oriented model only.) The DFD haves the representation of processes, data base, external entity, data-flows respectively by ellipse, open ended rectangles, rectangles and arrow marks between two entities. DFD goes into deep of the system representation as its level increases starting from level 0(generally DFD up to level 3 gets appreciated otherwise becomes too complicated and clumsy). The level 0 DFD is called context diagram which shows the interactions between the system and external objects/agents which respectfully act as data sources and data sinks. There is an establishment of system boundary inside which the whole system to be analyzed stays and outside the boundary stays the objects not to be bothered about. Context level Data Flow Diagram shows the overall functionality of the application as a whole, a black-box view. The same is divided into separate modules in level 1 DFD and each module gets further separated in detail in level 2 DFD and so on.

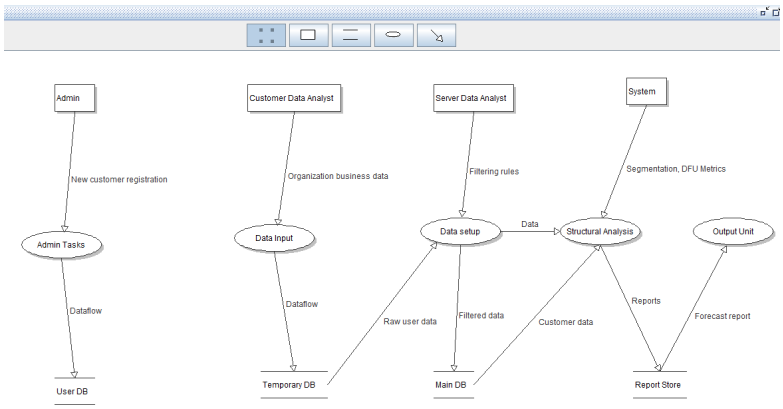

Figure 3.1: DFD example of an e-forecasting system

Hence the sub-systems are found from level 1 DFD and the subsection detailed views and further detailed views are found from level 2 and level 3 DFDs. System boundary is also named as trust boundary since it is the interface that carries the level of trust of the system. It is the area where security gets concerned, otherwise inside a trust boundary there can be no involvement of any external entity but only the process and data flowing through it. Figure5.1 shows an example of level-1 DFD of an e-forecasting system. There are four external entities Admin, customer data analyst, server data analyst and system present. Four processes are present named as admin tasks, data input, data setup, structural analysis and output unit. Four databases are there : user db, temporary db, main db and report store. The dataflows among them is shown by labeled arrows.

The threat modeling process is a step by step process and is best described by Figure3.2.

The stages of threat modeling process are shown in Figure3.2 and are starting from business objective, identification of security objectives, system overview, decomposition of system, identification of threats, identifying security controls, risk analysis and remediation and again going to system overview stage following an iterative approach for further refinement. The complete process is explained through a trivial example of student grades display system.

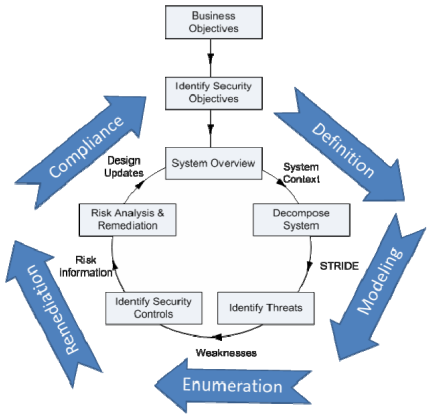

Figure 3.2: Threat modeling Step by Step process

1. Business objective

This stage defines the business goal of the organization, schedule for completion of different phases of SDLC, monetary analysis and requirement gathering for the software. In the trivial student grades display system, the business objective may be simply to build an application that can show the grades of the students to the teacher requesting to see it. Since its a small software with a single functionality, no mention of schedule, monetary analysis or anything else is needed.

2. Identifying security objectives

From the business objectives, the requirements are made apparent and taking both as basis, the security objectives are identified. The security objectives may be implicit or explicit. It is said to be explicit, if the system in its requirements has explicitly mentioned about the security requirement; and it is implicit if the designer himself assumes or derives the security requirement that has not been mentioned explicitly.

In the student grades display system, the security objectives can be like preventing the display of grades to someone other than teacher, prevention of illegitimate modification of the grades, uninterrupted service to the teachers for display of marks, allowing only teachers to get authorization to view the marks(as the secure system means, the system does not do what it should not do), not let even other teachers to modify the marks given by one teacher though they might

be allowed to see it, if possible maintain a log of activities of teachers of who is modifying which data. If the security objectives mentioned above are classified, they can be classified as confidentiality, integrity, authentication, authorization, accountability respectfully.

Table 3.1: Business and Security objectives of the Student Grade Display System

Business Objective	Security Objective	Property	Definition
BO1	SO1	Confidentiality	Prevent unauthorized Disclosure of students' grades
	SO2	Integrity	Prevent unauthorized modification of students' grades
	SO3	Availability	Always display student's grades to the teacher
	SO4	Authentication	Establish Identity of end user before allowing access to system
	SO5	Authorization	Teachers can see grades entered by other teachers but can not modify
	SO6	Accountability	Maintain audit trail for critical functions like student grade modification

3. System overview

In this phase, a complete system functionality from a broad view is taken and graphically represented. The presentation can be using any of the data flow diagram or the control flow diagram(uml diagrams). Microsoft approach takes the data flow diagram for this purpose. If data flow diagram is considered, this stage requires the level 0 DFD (also known as context diagram) to be depicted. This is needed to separate out the system from the external environment and find out the external entities and databases external to it interacting with it.

In this phase, the potential assets are identified. Each asset is an object which is subject to exploitation by the attacker. This system overview can be related

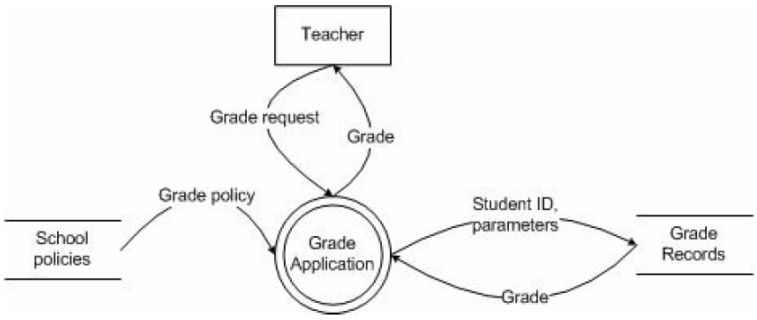

Figure 3.3: S
ystem overview of Student Grades Display System System overview of Student Grades Display System

with the security objectives on the assets whether implicit or explicit in this phase. The documentation is optional.

As in the student grade display system, the system overview diagram is the following context diagram Figure3.3. This Figure shows the teacher is the only external entity interacting with the system, as a whole, shown in a multiple process notation. It interacts with two databases: school policies and grade records.

The assets identified can be related to the security objectives and to the functionalities here and are stated in Table3.2 .

Table 3.2: Assets related security objectives and functionalities

Functionality	Description	Role	Data	Security Objectives
F1	When teacher enters valid student number, system displays the student grades. If the student number is invalid, system displays an error.	Teacher	Grade store	SO1, SO3
F2	Teachers can only modify the grades they themselves have entered	Teacher	Grade store	SO2, SO5

4. System decomposition

The overview system developed in the previous stage is now decomposed into small components for avoiding complexity in threat modeling. These decomposed

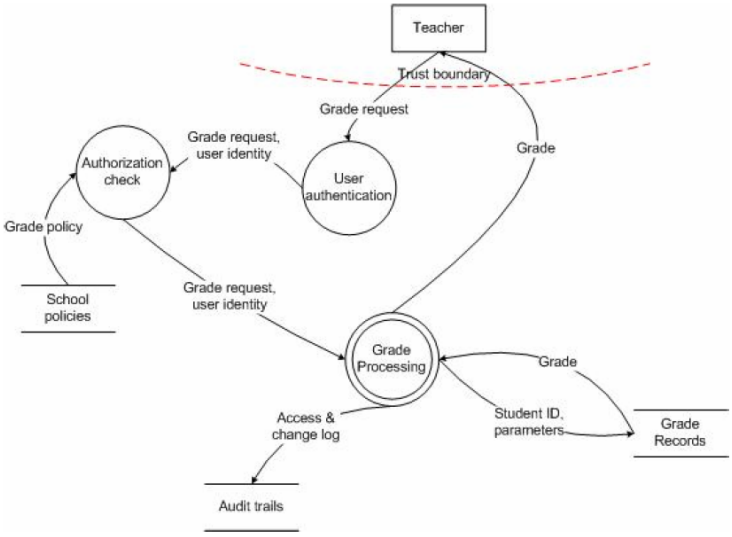

Figure 3.4: DFD of the Decomposed example System

systems are interconnected to each other via data flowing in and out of them. This divides the big task into interconnected subtasks. Introduction of trust boundaries comes here on the user interfaces at the boundaries of the system. These are the attack surfaces that may be exploited by the attacker and hence the goal of the designer is to try minimize the attack surface as much as possible. Less the attack surface less the attack.

There is another advantage of this decomposition and simultaneous introduction of trust boundary, ie the data flowing through(in and out) the trust boundary is to be probed while the data flows that do not cross these boundary need not be worried for.

The example system is further divided into components as shown below in Figure 3.4. The decomposed system is self explanatory, which introduces a trust boundary at the boundary of the system with the teacher.

To describe better realization about the security objectives, some more functionality is added and they must also be added to the functionality as shown in Table3.3.

Table 3.3: Table showing Additional Functionalities Added to the system

Functionality	Description	Role	Data	Security Objective
F3	Authenticate each user before allowing access to system	Teacher	Credential store	SO4
F4	Allow administrators to add more users to teacher role, edit and delete users from teacher role	Administrator	Credential store	SO4, SO6

5. Identifying threats

This phase practically associates each element of the software with all possible threats on that. Here each element from the decomposed diagram is taken into scrutiny and checked for each possible threat against that in the specific deployment environment. By not doing it smart and systematic, this phase is going to consume a lot of time if individual element against a large number of threats are considered. To avoid such problem, Microsoft proposes a methodology called STRIDE methodology to efficiently identify threats and that too systematically. The approach has been discovered by S. Hernath, S. Lambert, T. Ostwald and A. Shostack from Microsoft Developer Network(MSDN) and described in [22, 24]. The same approach has been followed in [25] by JP Jesan. This method works prominently against data flow diagram [26] and can be made to act with other modeling diagrams as well(it has been made to act with activity diagram and shows similar results).

STRIDE word is made from the initials of six different threat classes: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privileges. The methodology claims that any threat that may happen in a web application can be categorized into the above mentioned six different threat classes.

Table3.4 explains the six different categories of threats

The spoofing attack is simply fooling someone else by impersonating someone

Table 3.4: STRIDE Security concepts

Property	Description	Threat	Definition
Authentication	The identity of the user is established.	Spoofing	Impersonating something or someone else
Integrity	Data & System resources are only changed by intended people	Tampering	Modifying data or code
Non-repudiation	User cant perform an action and later deny it	Repudiation	Claiming to have not performed an action
Confidentiality	Data available to only intended persons	Information Disclosure	Exposing information to unauthorized person
Availability	System is ready when needed and perform fine	Denial of Service	Deny or degrade services to user
Authorization	Users are explicitly allowed or denied to access resources	Elevation of Privileges	Gain capabilities without proper authorization

else by gaining access to his privileges. This is the threat to the security property authentication. The tampering attack means modifying the specific data and code flowing between processes or residing in memory of the process or the database. This is a threat to integrity property of security. If someone says I have not done this work, anybody else has done or if he suggests someone else to have done some work and there is no proof of his statement with the system, that case is considered to be repudiation in the system. A non-repudiated system does have every activity information performed on it. Information disclosure is the case when some system reveals information to someone not intended to know which should not have been done. This is a confidentiality issue and compromises to the privacy of data. Sometimes the availability of the software to its legitimate user comes under threat. The functionality and resources of the system is heavily kept busy and thus the system slows down, denies for further serving or even crashes. Such a threat is called denial of service(DOS). In case this is done by a number of systems over number of networks(configuration known as botnet) to attack a server or machine the same is called as distributed denial of service attack. The last class of threat is elevation of privileges, happens when an anonymous user gets the privileges of a normal registered user or the normal user gets the privileges of admin who in turn can misuse their privileges. This is a threat to the authorization aspect of security.

This stride model is mostly applied to the data flow diagrams since a very useful

Element Type	Threat Types					
	S	*T*	*R*	*I*	*D*	*E*
External Interactor	×		×			
Process	×	×	×	×	×	×
Data Storage		×	×	×	×	
Data Flow		×		×	×	

Figure 3.5: STRIDE applied to Data Flow Diagrams

and efficient threat-element relationship establishes which is shown in Figure3.5.

The STRIDE model actually squeezes the scope of the designer to test for all possible attacks against a particular entity, rather it makes the design more efficient and accurate by confining it.

The external entity may be anything, it can be system or human. So any input sent by it is unpredictable. Hence tampering attack on external entity is not appropriate. The information present in an external entity is not a concern to us since it resides outside the boundary of the system. Hence Information disclosure does not apply to them. The same is the reason for the non-application of denial of service threat class to it. Again, since of an instance of an external entity is always constant throughout the interaction session of it with the system, so privilege escalation also does not make sense here. But since the external entity is a human being or any automated system, it can be spoofed/ fooled or it can be subjected to repudiation. The Process is subject to all classes of attacks.

The data storage cannot be spoofed as it is permanent(though showing false names of data stores humans on external entity can be spoofed). Since it does not operate in any privilege level, the last type of threat also is not reasonable on the data stores. Rest four classes of threats can be applied on data storage. But the repudiation attack on the data storage needs special attention, because it happens only when the database maintains a log and it is being manipulated by outsiders.

Data flows cannot be spoofed nor be repudiated. Again there is no privilege access to the data flows.

In our example of simple student grade displaying system, the external entity teacher is subjected to SR classes of attacks, the process user authentication,

authorization check and grade processing processes are subjected to all 6 types of threats. The school policy, audit trails and grade store data base are subject to TRID types of attacks, and the data flows in between all the elements are subjected to TID types of attacks.

6. Identifying security controls

The various components are to be evaluated against each threat. It is checked whether the data flow is through the trust boundary or not. If yes, it has to be brought under scrutiny as it is exploitable to threats. If not, on mitigation of threats at the boundary level entities, it gets automatically mitigated. The vulnerabilities can be technical ones eg sql injection, cross site scripting or logical vulnerabilities eg. Absence of appropriate checker/ validator or specification vulnerabilities for example allowing easy-to-guess passwords. Brainstorming tasks have to be done to identify the type of the vulnerabilities on the particular asset at a specific point under particular situation and a proper identification of mitigation to eliminate the vulnerability.

In our case of simple student grade display system, teacher is subject to spoofing and repudiation attacks. Let us take a case, the teacher requests for grade to the system and receives the student grade. As per the security objective SO4 stated above, the teacher must login to the system before accessing it. To achieve these functionalities, now the designed must analyze the aspects like the options that can be implemented for authentication whether form based or certificate based or operating system integrated. If a form based authentication gets selected, what are the options for accepting the password, storing, updating or renewing them. Again, where are the storing options? LDAP or simply files or will it be a database. If LDAP is a choice, which product should be used? Whether active directory or open LDAP. If active directory is considered, whether to take the cost of buying a windows server license? Again, when sending the first time credentials, whether it should be sent over mail or SMS or by post? Whether to send it to the teacher or to the principal or should it be an online registration? All these types of brainstorming activities have to be done. After it gets confirmed on the answers of

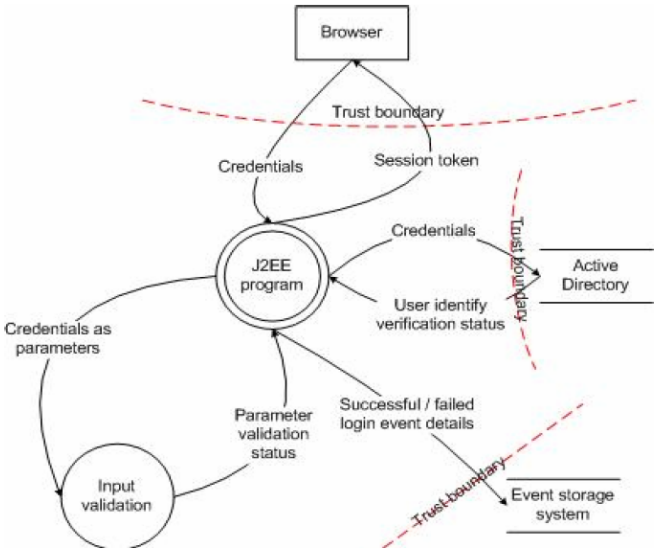

Figure 3.6: Authentication Section Decomposed in the example System

these questions, the entry points start to be taken care of and vulnerabilities in assets and the attack paths start to be identified.

Going into a little detail level, suppose the example system is designed using J2EE with a form based authentication and the authentication information is stored in a active directory. The authentication module of the system can be presented as Figure3.6.

From Figure3.6, its time to analyze the specific threats on the elements. We start with the sql injection and session parameter hijacking. It can be considered as a spoofing threat if there is no proper input handling and input validation. In this case, the attacker can bypass the weak authentication scheme and because of the improper session parameter handling, attacker can lunch any type of attack like session prediction or session fixation. So now the advantage of the threat modeling can be realized: even before writing a single word of code, it has been known that there is a chance of SQL injection and the input validation has to be taken care of and better not to allow any special character to the input field of form-based authentication page.

7. Analysis of Risks and remediation against them

Once the threats and vulnerabilities are found out, then the designers go for analyzing, prioritizing and making decision on the mitigation plans. It is not efficient or economic to address all the risks because some of them have very less chance of occurring, or some of them have very less damage potential. So the designer goes for prioritization. For prioritizing risks, a score is assigned to each threat or a risk score is assigned to a combination of vulnerabilities to find out severity and support for decision making. The basic equation for decision making is given by

Risk score = Probability of occurrence * Business impact

Microsoft has developed an approach DREAD for rating the risk. The approach of Microsoft has been proposed in [23] which has been adopted as standard by OWASP community and has been described in its Journal [27]. The same approach was followed by K Ram Mohan Rao and Durgesh Pant for a case study of Geospatial Weather Information System (GWIS) for the purpose of threat risk modeling [28]. This DREAD is mentioned in detail in the following texts.

DREAD is a word made from 5 different words initials. They are: Damage potential, Reproducibility, Exploitability, Affected users, Discoverability. Damage potential defines how much damage to the system can occur once the vulnerability has been exploited. It shows the measure or extent of it. Reproducibility defines the easiness of execution of the attack and repeating the attack. Exploitability defines the easiness of lunching the attack and what amount of expertise is required for an attacker to launch an attack. Affected user shows what number of end users get affected by the exploitation. Discoverability defines the easiness to attack the system or find out the vulnerability.

Each of these values is affected by the primary security parameters: confidentiality, integrity, Availability and Accountability.

Damage potential and Affected users contribute towards the business impact, while the rest three Discoverability, Exploitability and Reproducibility contribute to probability of occurrence. So re-writing the formula,

Risk Score = (Discoverability + Exploitability + Reproducibility) * (Damage potential + Affected users)

Additional considerations will include the extent of exploitation, automated attack and pre-existing mitigation controls. Now a number needs to be assigned to each of the above mentioned factors so that the Risk score can be numerically calculated. Below mentioned are the conditions on which the scores are calculated to be high, medium or low. On a scale of 10, 10 is assigned to the high value, 5 to the medium and 0 to the low value.

Damage potential: The critical value is used when damage impacts a lot more like the existence of the company or the high value assets are impacted or loss occurs to them, medium value is assigned when some valuable tangible assets are damaged and low value is assigned when the damage is minimal.

Reproducibility: the high or difficult value is assigned when the vulnerability exploitation is difficult to repeat even if the attacker is provided with the knowledge of the architecture of the system; medium value is assigned when given some given conditions and situations the attack may be made to happen and low value is assigned when the system can be exploited any time.

Exploitability: Difficult value is used when the attacker, even after getting the knowledge of system internals cannot launch an attack. Medium value is used when the attack can happen given certain conditions and low value is used when exploitation is possible all the time.

Affected users: high when the affected users are many, medium when the numbers of users are moderate and low when it is very less.

Discoverability: High value when the vulnerability is very trivial and any user can exploit it without having much knowledge of the system architecture. Medium value is used when the exploitation of the vulnerability is known to few and it needs some brainstorming exercise to exploit it and the value is assigned high when the vulnerability discovery is too hard to do.

Table3.5 shows the values of severity of the 5 factors of DREAD.

Based on the above scoring system, the risk mapping can now be prepared.

Table 3.5: Values of DREAD

Attribute	Values
Damage Potential	High(10), Medium(5), Low(0)
Reproducibility	High(10), Medium(5), Low(0)
Exploitability	High(10), Medium(5), Low(0)
Affected users	High(10), Medium(5), Low(0)
Discoverability	High(10), Medium(5), Low(0)

Probability of occurrence = Reproducibility + Exploitability + Discoverability

So maximum probability of occurrence= 10+10+10=30 Minimum probability of occurrence = 0+0+0=0

Business impact = Damage potential + Affected users

So maximum damage potential=10+10=20 Minimum damage potential = 0+0=0

Now risk score = business impact * probability of occurrence

Maximum risk score = 30*20=600 And minimum risk score = 0 And medium risk score = (5+5+5)*(5+5) =150

So by this it may be a measure like, a threat with risk score in the range 0 to 100 can be taken as a low risk threat. 100 to 300 risk scored threats can be medium risk threats and 300 to 600 risk scored threats to be high risk threats and they have to be mitigated at first. Here a variation of 50 is taken down-way and a variation of 150 is taken on up-way calculation for the medium risk.

This calculation may also be done using a scale of 1 to 3 and it shows a reasonable illustration as well.

After the risk has been calculated, the designer goes for remediation of the threats from higher priority ones to the lower priority ones.

• **Remediation against various classes of threats**

Spoofing: Spoofing, as has been already defined, is the process of impersonating someone to get his privileges and misuse the same. Spoofing is of different types like IP spoofing, URL spoofing, Email spoofing etc.

IP Spoofing: It comes under spoofing the external entity. It is the way of getting the unauthorized control to the computers and letting the trusted computers(which are under attack now) to send message to another system making an illusion that the sending system is a legitimate one. The IP address is forged. The IP spoofing attack may be of many types like

1. Non-blind IP Spoofing, in which the attacker stays on the same subnet as that of the target system and can see the sequence number and acknowledgement number of the packets. So it is easy for him to interfere with a connection to send packets along the subnet. Example of this kind of attack is session Hijacking.

2. Blind IP Spoofing, in which sequence number and the acknowledge numbers are unknown , hence he has to predict or send arbitrary packets to a system to gain this information. Its too difficult to carry out on present date.

3. Man in the middle attack, in which the intruder may gain access to the legitimate message transfer between the sender and receiver and can send his own message to both pretending to be the other trusted party. Ie, the receiver thinks the message is coming from the sender and vice versa. This type of attack is also termed as connection hijacking. Man in the middle attack can be prevented by introducing nonce into the messages and encrypting the message.

4. Denial of service attack, which is caused when the attacker spoofs someones ip address and floods the receiver with heavy request for resources and operations.

5. SMURF attack: its performed by flooding ICMP packets with spoofed IP addresses to a LAN which later broadcasts the same to all other hostson the LAN. As a result, all other hosts send reply to the spoofed ip address which gives rise to Denial of Service.

The IP Spoofing attack is detected by monitoring traffic within the organization with the help of the network monitoring tools like Netlog. A packet which is outside the domain and is having the source and destination address of the local systems, is considered to be a packet with IP Spoofing. In that case, if a spoofing attack happens, the system with the IP address, which is used by the attacker, shows a trace of remote access in its activity log. So by comparing the activity

logs of all local systems, the compromised system can be found out. Another way of detection is source address validation of the packets by the router.(Generally routers are only concerned about the destination address only).

Prevention of IP spoofing attack needs first of al installation of an access control list(ACL). This list is a list of permissions that is attached to an object. An ACL specifies the access granted as well as operation allowed on particular objects by particular users. A filtering router can be installed that restricts the input to the external interface(known as input filter) by not allowing a packet to cross it if it has a source address from the local network. Also the outgoing packet can be filtered with a source address other than the local network address, so that the IP spoofing attack originated from local network can be prevented.

URL spoofing: this can be treated as a process spoofing attack where one web site imitates another one and gets sensitive data from the user. Intrusion is the attack caused by it.

Security patches that are updated by the popular web browsers update the feature of un-masking the "true" URL of a site in the web browser. It also shows the security certificates issued to the website and lists down the insecure ones. So its important to stay updated to the new updates. Email spoofing: this type of attack happens the header of an email so that the email is altered by the attacker so that it seems as if sent from someone else. Hence it may cause confusion, discredit the person or hides the identity of the sender. The prevention of this problem can be done by checking the header of the email.

Remediation suggestions for Tampering in processes:

1. A process is said to be tampered if its bits are changed while in execution. So integrity controls and careful validation of input has to be done.

2. Input validation by maintaining appropriate white list.

3. If the callers of a process are given the access to the shared memory/pointers or are given ability to control what is executed (for example passing back a function pointer), then they may tamper with the process. So here comes the trust issue on them. better give least access to memory. Better to pass data instead of pointers.

Validation has to be performed.

4. The programs the process is calling for, has to be secure. We should use fully qualified path names to call them and maintain ACLs for the opened objects.

5. Shared memory, used for inter process communication, should be used with ACLs. Compromised process sharing memory may inject any type of malicious data to the shared memory. Hence the shared memory contents have to be validated before processing them.

Remediation suggestions for Tampering in data flow:

1. Tampering in data flow is altering bits on the wire or between two processes. Cryptographic integrity control for the data in network and good use of the operating system features to protect the inter process communication from the tampering of data bit of it has to be done.

2. An anti-replay technique and a strong integrity technique has to be followed. It is because data flows without sequence numbers or timestamps can be captured and they may be replayed by the adversary. So the dataflow has to be defended by the hashing or MAC or digital signature technique and time stamp or counters have to be followed.

3. Duplicate or overlapped data in the data flow are checked for. For example, let packet 1 is of 100 bytes and it starts with offset 0. Packet 2 is also of 100 bytes starting at offset 25. In that case, packet 2 overlaps the packet 1 and the packet 1 looses 75 bytes of its data. So the system has to ensure this fact at the re-assembling time that packet should not be overlapped neither it should allow duplicate packets.

4. To prevent the man in the middle attack, the end points should be authenticated to each other before the start of the session. So a key persistence algorithm (a la SSH) or a PKI (public key infrastructure) or Kerberos has to be implemented. If the system is not using one of these three, review has to be done on the design whether man in the middle attack is not a high risky matter in the system.

5. Standard protocol like SSL has to be adopted for a strong message integrity

system.

6. The keys have to be kept secret for a strong channel integrity system.

Remediation against tampering in database:

1. Integrity control and careful input validation. Tampering with a database involves changing the bits or causing the different bits to get returned.

2. Set permissions properly. Give least required permissions to each user, even to admin.

3. The program has to be ensured to be the only thing to access the database and the users or any other system can have the API only to interact with the system and there by the database.

4. Proper validation to the inputs should be done and special character entry should be carefully validated.

5. Error should be returned in case database is full. This case arises when either the data is discarded when full or the data storage wraps to its start when it is full. Again, the error should be a generic one and it should not reflect the details of the internal process.

Remediation against Information Disclosure in process:

1. Encryption has to be considered for the process memory along with the keys and the process memory should not be storing any secrets that are no longer required for process execution, since process memory may be read by the attacker.

2. Using a white list of legitimate input symbols, input validation must be done for the user inputs to the processes.

3. ACLs should be protecting process memory and validating shared memory by its mentioned access to the legitimate users only.

4. A side channel analysis has to be performed to prevent the side channel attacks. Constant time approach should be applied to encryptions to increase the chance of un-ambiguity in the encrypted message to prevent side channel attacks.

Remediation of Information disclosure in database:

1. The database data should be considered for encryption and the access control list should be introduced for the access of the database.

2. If there are other consumers to the database, there is a chance that they may bypass the protection mechanism. To protect against that, APIs should be provided to access data through reference monitor.

3. The unprotected objects or tables in a database should be listed down and kept an eye on that. Appropriate security policies should take care of it.

4. The program has to be ensured to be the only thing to access the database and the users or any other system may have the API only to interact with the system and there by the database.

5. A side channel analysis has to be performed to prevent the side channel attacks. Constant time approach should be applied to encryptions to increase the chance of un-ambiguity in the encrypted message to prevent side channel attacks.

6. In case of an undo or recovery, the files should be cleaned explicitly and its rarely a performance issue.

Remediation of Information disclosure in data flow:

1. The data should be considered for encryption and the access control list should be introduced on the system to access it so that it is not going to be used by anonymous users.

2. The key exchange or key validation should be performed out of band to ensure that all end points are mutually authenticated.

3. Encryption should be done to the messages as it crosses the network. A strong channel confidentiality system and message confidentiality system should be ensured.

4. A side channel analysis has to be performed to prevent the side channel attacks. Constant time approach should be applied to encryptions to increase the chance of un-ambiguity in the encrypted message to prevent side channel attacks.

Remediation of Repudiation in External entity:

1. The user activity should be logged.

2. Standard digital signature scheme should be introduced.

3. An anti-replay technique and a strong integrity technique has to be followed. It is because data flows without sequence numbers or timestamps may be captured

and they may be replayed by the adversary. So the dataflow has to be defended by the hashing or MAC or digital signature technique and time stamp or counters have to be followed.

4. Only trusted code should be allowed to be logged. If the system starts to log for every anonymous user, there is a chance that DOS may arise.

5. Sufficient space should be there for the activity log so that it does not run out of space.

6. Log readers may come under attack via log files. So ways to canonicalize data in the logs should be thought of. Long user input may be truncated and if there is a single reader for the logs, they should made to know where each field comes from, and which are untrustworthy. If many readers are there, the dos and donts should be documented so they can figure out where each field comes from.

Remediation of Repudiation in Process:

1. The process should be made to run at a lower privilege level. The logs of the process should be handed over to a process run as a different user(which can act as trusted party). It is because once a process is compromised, it cannot provide non-repudiated data. It happens by the process, getting the admin privileges, may destroy the log files or audit data created before the compromise.

2. Standard digital signature scheme should be introduced.

3. An anti-replay technique and a strong integrity technique has to be followed. It is because data flows without sequence numbers or timestamps may be captured and they may be replayed by the adversary. So the dataflow has to be defended by the hashing or MAC or digital signature technique and time stamp or counters have to be followed.

4. Only trusted code should be allowed to be logged. If the system starts to log for every anonymous user, there is a chance that DOS may arise.

5. Sufficient space should be there for the activity log so that it does not run out of space.

6. Log readers may come under attack via log files. So ways to canonicalize data in the logs should be thought of. Long user input may be truncated and if

there is a single reader for the logs, they should made to know where each field comes from, and which are untrustworthy. If many readers are there, the dos and donts should be documented so they can figure out where each field comes from.

Remediation of Repudiation in Databases:

1. Logs in database should be made to contain enough data to analyze things after even the deletion of the logs. So they stay well protected.

2. Strong access control lists should be used.

3. Enough data should be captured in the log.

4. Audit logs are more susceptible to come under attack. So access to the logs should be ensured through a reference monitor, which can control read and write separately.

Remediation of Denial of service in process:

1. The applications that take inputs from users on network are considered and how much processing is done against each of the requests and how much CPU and resources are utilized for each of them should be calculated. Accordingly a limit or rating parameter should be introduced and/ or authentication for users should be considered.

2. A white list should be maintained for validating all inputs.

3. Resource consumption attacks are to be dealt with. Better let the OS do the job. It is to be ensured that the resource requests do not deadlock and that they do timeout.

Remediation of Denial of service in database:

1. It happens when data involves either deleted or has run out of space. That is, if a data store is either full and it is tried to be written or unexpectedly empty when it is tried to be read. For it permissions should be properly set.

2. The data store names may be hijacked or squatted. So all names should be hard to predict and well protected. The file system should not be shared and the registry access across different trusted parties should not be shared.

3. The app should deal with an unavailable data store to make fool to the attacker. Log for that false data store should be kept also.

4. Resource management should be done.

Remediation of Denial of service in data flow:

1. On the system, it has to be ensured that not everyone can change the ACL. On the network, it is to be ensured that anonymous users do not use up resources.

2. Strong authentication scheme has to be considered.

3. The file system should not be shared and the registry access across different trusted parties should not be shared.

4. SSL should be used for transport layer level strong integrity and authentication control.

Remediation of elevation of privileges in process:

1. Careful validation of all user input for appropriate purposes otherwise it may lead to buffer overflow.

2. To validate all inputs, a white list to be maintained.

3. If shared memory is used for Inter process communication, ACL should be carefully used. Also it should be kept in mind that if other processes sharing the memory are tampered with, it may write malicious things to the shared memory. So the data from the shared memory should be validated like all other user inputs.

After the successful completion of all the above phases, iteration of all the phases from the beginning is done for any further refinements on the system before finalizing the threat modeling documentation.

Besides the above mentioned process of Microsoft and all its methodologies, there are some other techniques used in the threat modeling process like attack tree, misuse case diagram etc. which are explained below.

3.1.2 Attack tree:

Attack trees can be defined as a formal way of expressing the security of systems, based on scalability of attacks, through a tree representation. Attack tree was introduced by B.Schneier which claimed to be capable to be used in divergent fields [29]. Threat modeling using the attack trees was done by V. Saini, Q. Duan and V.Paruchuri and was described in [31]. I. Morikawa and Y. Yamaoka claimed that use of attack trees make the difficulties in threat modeling rather easier for

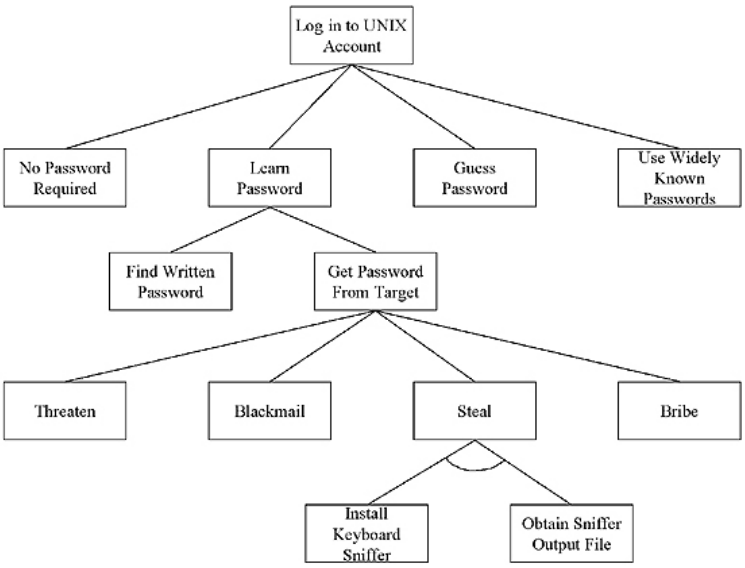

Figure 3.7: Example of attack tree: Login into UNIX

non-experts in security by introduction of attack tree templates, as explained in [30]. The tree construction goes as this: The tree haves the goal of the attacker as its root node while the ways of achieving the goal in its leaf nodes and it continues in a top to bottom fashion. The children to a node are connected to each other by and or or nodes. And nodes define the children nodes with the and nodes combining together achieve the goal while the Or node represents the alternative of the associated ones. Figure3.7 shows an example of attack tree representation of logging into the UNIX Account .

In Figure3.7 the primary goal of the attacker is to login into the UNIX account which can be done by wither of the following ways: by not entering password, or by learning password or by guessing the password or by using the common widely known passwords. If the learning the password way is considered, it may be done either by finding out the written password or by getting the password from target machine. The latter may be achieved by either threatening the owner of account, blackmailing, bribing him or by stealing the password. The password may be stolen by installing keyboard sniffer and obtaining the same sniffer output file. This one is the AND node and all others are OR nodes. Attack trees are not only useful in representation, they are also useful in the Oil/gas pipelines, Chemical

Plants, infrastructure, and Facilities etc. they are very useful in the field of risk analysis.

The attack trees are useful in security representations because they are most flexible for the scalability of the attack profile. Only the leaf node needs to be updated since the goal, which in the internal node, is the same for an attack. Another advantage of using attack trees is the bottom to top propagation of scores or values or cost or mitigation suggestions. In this representation, all the top nodes may be evaluated by evaluating the leaf nodes only. This feature actually makes it very useful in many fields like risk analysis.

3.1.3 Misuse case diagram:

Misuse case, also termed as abuse cases from a broad view (though some minor differences exist as stated in can be defined as an evolution of use case which describes the behavior that the system or external entity does not want to occur. The misuse case description was modeled as a template by G. Sindre and G. A. Opdahl in [34]. The conflict between the legitimate operations and the malicious operation in a business organization was first shown by I. F. Alexander in [33]. Elicitation of security requirements was formally done as a vital use of the Misuse case diagram and shown by Sia Chun Wei in [32] and by G. Sindre and G. A. Opdahl in [3]. The Misuse case diagrams, sometimes referred as abuse case diagrams are basically used to show the malicious activities acted upon the functionalities of the actor just like the definition suggests. Figure below 3.8 shows the misuse case diagram for a simple order processing system, which is very much self-explanatory.

The misuse case diagram, used to show the malicious activities, is acted upon the use case diagram, but in an inverted manner (shown in black color). There is one or more than one misactor identified for each actor in the use case diagram. All the malicious functionalities done by the misactor are depicted in the same diagram where the legitimate functionalities lie. So if the threat modeling has to be done upon the control flow of the system, this misuse case diagram can be a handy figure to initiate things.

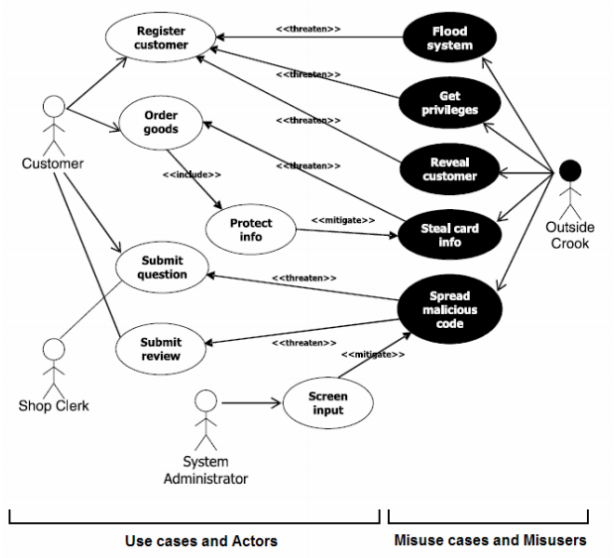

Figure 3.8: Misuse case example of a simple order processing system

3.2 Tool support for threat modeling:

the whole process of threat modeling can be roughly conducted without using any support of tool or any particular framework since the main concern in the whole process is the brainstorming of threats possible on an application. But given the huge extent and coverage of the field, a systematic process to do so with step by step guidance and a structured report generated out of whole process can be productively helpful for many users. There are several tool support for the threat modeling activities that include TRIKE from Microsoft(defined and working on its own framework and approach), OCTAVE (an academic tool following its own methodology developed at the Carnegies Mellon University with the collaboration of CERT), some government aided systems such as Common Vulnerability Scoring Systems(initiated by the United States department of Homeland security) , Microsoft TAM tool (threat analysis and modeling tool) (being developed by the Application consulting and Engineering team of Microsoft), Microsoft SDL(security development lifecycle) threat modeling tool following its own SDLC life cycle etc. Recently, proposed by Asoke K Talukder et al at National Institute of Technology, Karnataka has developed a new open source tool supporting the hybrid approach stated in [9], named as Suraksha [35].

3.3 A Hybrid Approach to Threat Modeling

As it has been seen, there is some shortcoming in every approach of threat modeling. The asset-centric approach doesn't give any information about the attackers goal. This approach alone can hardly be used for threat modeling purpose. Moreover it doesn't prioritize threats. The attacker-centric approach needs prior deployment pattern knowledge and resources very clear from the beginning. This makes this approach ideal for clearly defined applications with very specific aim, not applicable for agile methodology. Software-centric approach finds its usage during the complete design and development phases, but for maintenance phase this approach doesn't provide appropriate solution. There is a need of an approach which can prioritize assets according to their risks, show all types of threats along with their priorities during development and after deployment the threat model can also be useful for the remedies against the scalability of wide range of threats. This is the hybrid approach for threat modeling. It makes the objective to take all the good sides of the above three approaches and in order to improve the efficiency and cost-effectiveness, prioritize the threats according to the risks of them on the assets. In paper [9], the authors have proposed a hybrid approach which comprises of the following steps: 1. Identification of Assets 2. Functional Requirements 3. Security Requirements 4. Threat and Attack Tree 5. Rating of Risks 6. Decision on In-vivo Versus In-vitro 7. Nonfunctional to Functional requirement 8. Iterate

1. Identification of Assets:

Assets are the reason threats exist in a system. An adversary does have the goal is to exploit the system by gaining access to the assets. The security team needs to identify all the assets of the system by organizing meetings, examining various documents and many brain storming sessions. The values of the assets are calculate by applying STRIDE and CI5A methodology and assign each with a value from three views : attackers view, Administrators view and users view. The valuations are also done from three aspects of security properties: confidentiality, integrity and availability. At last all the values of each resources for different

aspects of security are added and the ones with highest sum are taken to be the high valued assets.

2. Functional requirements:

in this phase, the functional requirements of the system are identified and modeled using use case diagram.

3. Security requirements:

for each actor in the use case diagram, misuse actors are created which may be one or more than it. They are analyzed for all types of possible attacks by application of STRIDE threats to each asset and for each action. This gives a list of many possible threats which is shown in the misuse case diagram.

4. Threat tree and attack trees:

Each threat in the misuse case diagram is considered as the root node of an attack tree which is considered to be the goal of the attacker. The attack trees are constructed for each and every threat mentioned in the misuse case diagram which represent the actual threat.

5. Rating the risks:

Using the DREAD methodology, the risks are prioritized and rated in a scale of 0 to 10. This is shown in the attack tree.

6. Decision in in-vivo vs in-intro:

In this phase, the priority of the threats are utilized to get the order of threat mitigation and to find out what threats may be left as they are by comparing with the prioritized assets listed in phase 1.

7. Non-functional to functional requirements:

In this phase the threats which are listed on higher priority after comparing with assets in the previous step are taken into the list of functional requirement(security is at first taken into non-functional requirement by default).

8. Iterate:

The above 1 to 7 phases are again iterated to check for some more refinements in the design before deriving a conclusion of threats.

In order to support the above hybrid approach, a tool was designed by the authors G. Santhosh Babu et al, named Suraksha and it had two workbenches [35] , one for developers and another for testing. The tool is available in the website isea.nitk.ac.in/suraksha.

Chapter 4

Threat modeling in live web applications

Threat modeling has been implemented on q web applications, which are getting developed at Tata Consultancy Services, one of the prominent IT companies in Asia. The threat modeling of the web application which is a scientific forecasting system, has been presented and described in details. Microsoft's SDL tool for Threat modeling, which is used widely for industrial projects has been used to support the threat modeling of the following application.

4.1 Threat modeling of scientific forecasting system:

The threat modeling of the scientific forecasting system has been explained in detail. The software is a live software currently being developed at Tata Consultancy Services, Bhubaneswar. The high level business objective of the system can be defined as the system takes the historical business sales data from all its registered organizations as its input, by application of different rules and statistical analysis, it produces the fore-casted report of future sales and demands as its output. The context diagram of the system is shown as Figure4.1

The system is associated with three different database: main database, staging database and temporary database. The biggest one out of them is the main data base which have the capacity in hundreds of Terra bytes. The customer sends business data and request to the system and gets his forecasted report back from

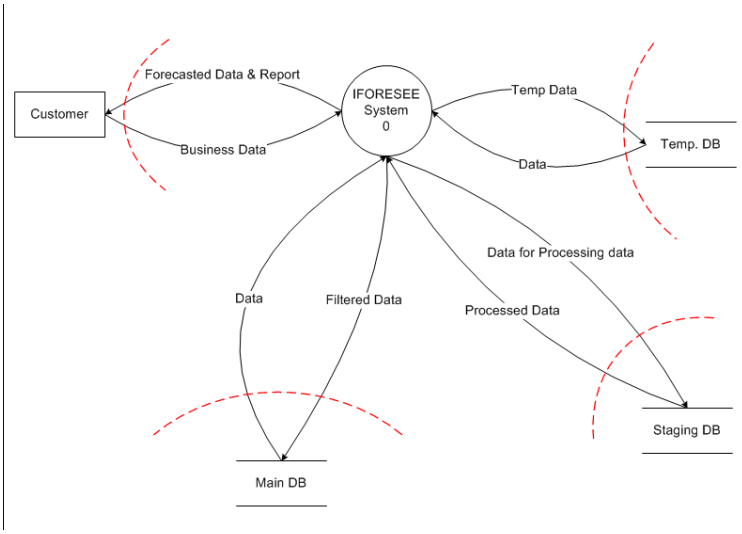

Figure 4.1: Context Diagram of Scientific forecasting system

the system.

After this stage of finding out the business objective, it is time for finding out the security objectives. This is a onetime activity where all the security concerns of the system are listed down and documented. In this software, the security objectives can be briefly stated as

1. The registered SCM user only should be able to upload and view the forecasted results. Any unauthorized user should not be able to do the same.(satisfaction of Confidentiality property)

2. No one other than the designated SCM person (SCM planning manager here) should be able to modify the output by the system.(satisfaction of Integrity property)

3. The system should provide uninterrupted service to the registered users.(satisfaction of Availability property)

4. Identity of the user should be established (preferably by session parameters) before allowing access to the system. (satisfaction of Authentication property)

5. No other SCM should be able to see the confidential business data neither the output of other SCMs. (satisfaction of Authorization property)

6. There should be a proper log maintained by the system which may be referred to in future on any modifications of the report done by the SCM plan-

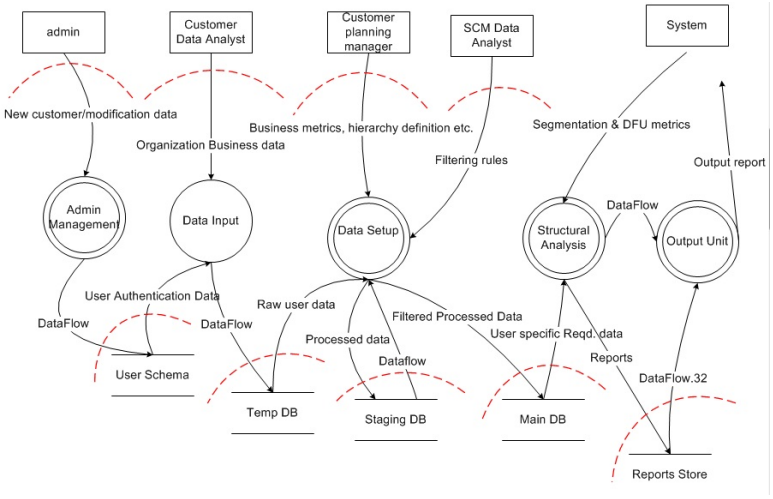

Figure 4.2: Level 1 DFD of Scientific forecasting system

ning manager and for all the transaction histories. (satisfaction of Accountability property)

These overall security properties have to be satisfied though out the development process and the end product should be satisfying the above mentioned six security objectives.

The overall functionality and its architecture can be shown on a level 1 data flow diagram as Figure4.2. This diagram satisfies the system overview identification of threat modeling process.

The actors interacting to the system are the admin, customer data analysts, customer planning manager, SCM data analyst and the system. The registered users are called Supply chain management (SCM). Each actor is assigned with some tasks which interact with different modules of the system. The admin is assigned with administration of the users and accounts and accesses, the data input module is handled by the customer data analyst who inputs the historical sales master data to the system. The data setup module pre-processes, filters, the data input by the customer data analyst with the help of the filtering rules defined by the SCM data analyst. In this module, the planning manager from the customer side defines different rules for forecasting like business metrics, hierarchy definition etc. In the next stage the actual statistical analysis occurs where the system while defining the segmentation and DFU metrics forecasts the demands

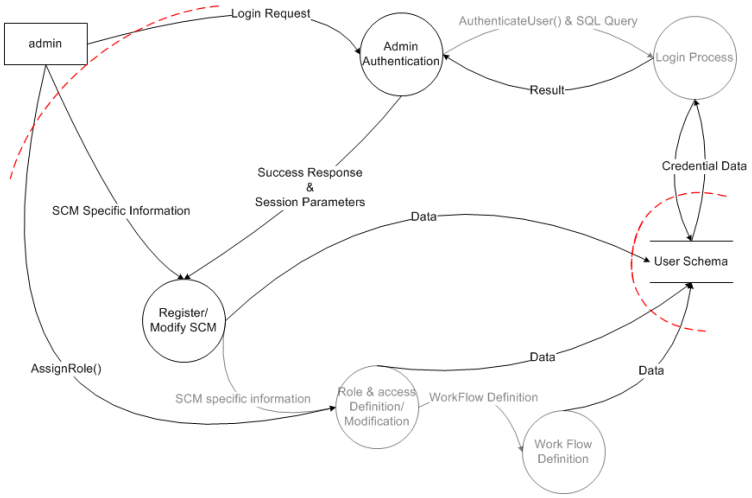

Figure 4.3: Admin Module

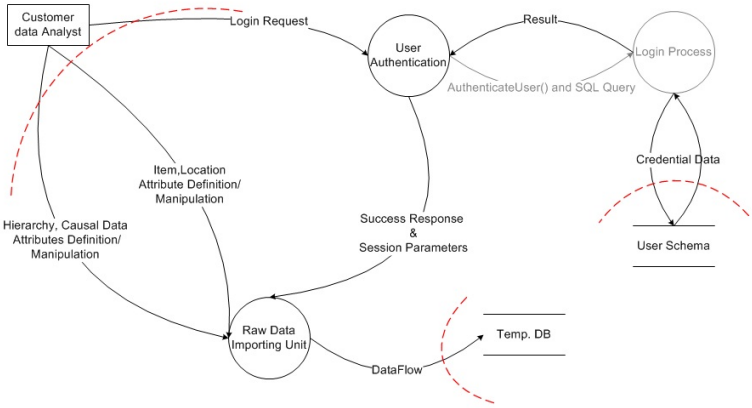

Figure 4.4: Data Input module

and sales for future one month or so. The output of the structural analysis goes to the output unit where the report is shown to the respective SCM data analyst.

Now its time to decompose the whole system into small interconnected modules. The decomposed system is displayed through its data flow diagram below.

1. Admin module: Shown in Figure4.3
2. data input module: Shown in Figure4.4
3. Data setup module: Shown in Figure4.5
4. Structural analysis: Shown in Figure4.6
5. Output module: Shown in Figure4.7

After the complete decomposition of the system each module is applied the

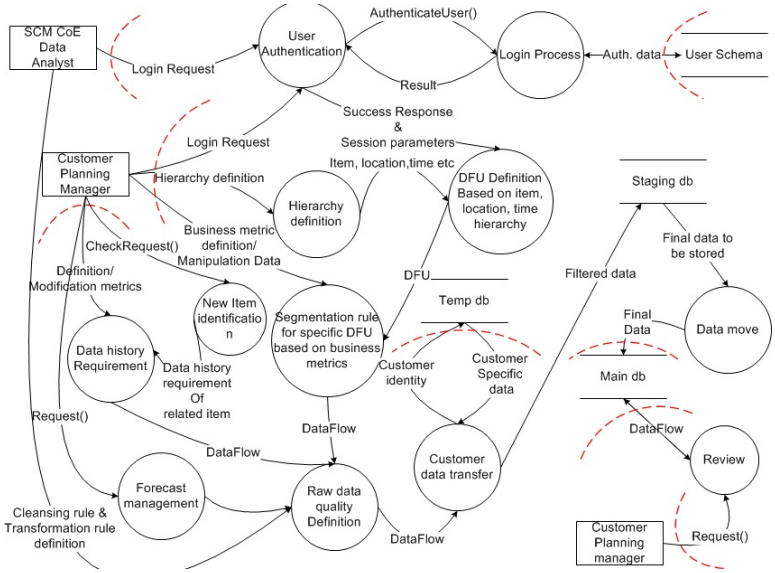

Figure 4.5: Data setup Module

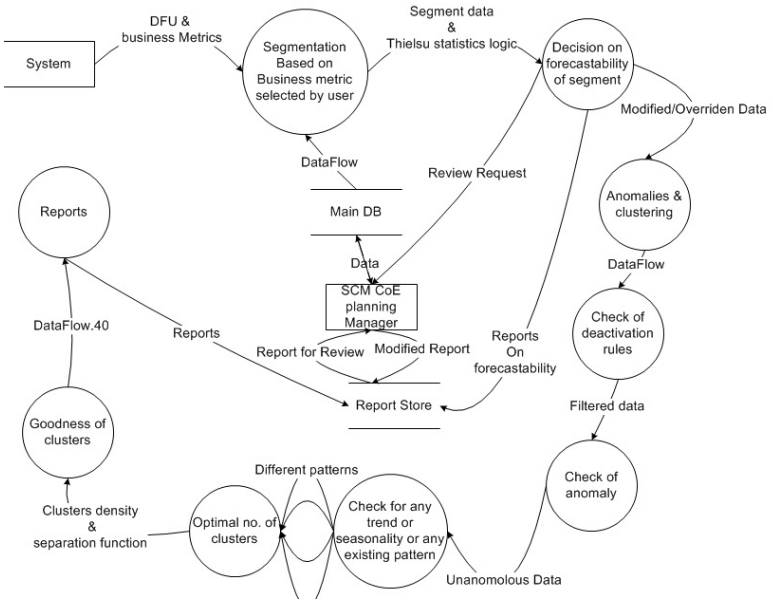

Figure 4.6: Structual Analysis Module

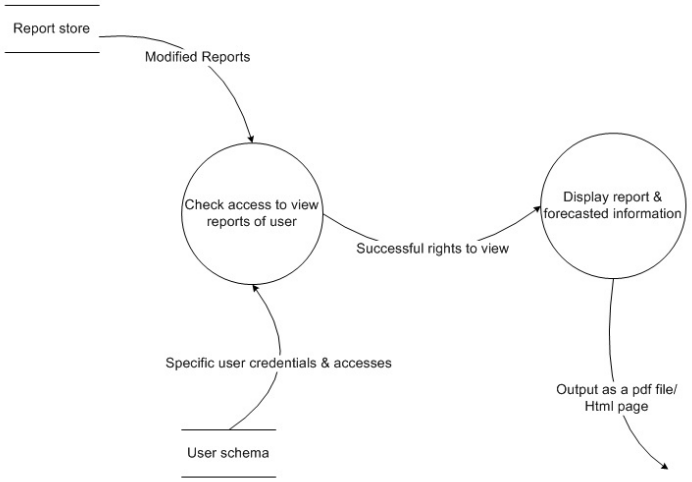

Figure 4.7: Output Module

STRIDE model to each entities. This is regarded as the identification of threats phase of the threat modeling process. By the application of STRIDE methodology, the possible types of threats on each entity of the data flow diagram can only be derived in a static way, but the threat class is subjected to shrink depending upon the deployment environment and situations. Once the STRIDE methodology has been applied, now it's the job to check what are the practically possible threats on the individual entities.

So starting from admin module, the threats to each element are analyzed here onwards.

Admin module:

The admin is the only external entity here, it is subject to spoofing and repudiation attacks. By session fixation or session prediction adversary may gain the privileges of the admin and ill-function his privileges. Also if the message transferred between the system and the admin is not properly encrypted and can be read by the adversary, he may launch man in the middle attack and fool the admin. Even in the simpler and worst case, the credentials may be stolen if it is stored with him in a text file or it can be guessed by an adversary. Without the appropriate maintenance of log, confusion might happen if the admin latter denies to have done some activity, leading to repudiation.

There is the only database which is the user schema database. The database is accessible for SQL injection by which data tampering and information disclosure may result. Side channel analysis can also be a cause for both the threats. Repudiation may result if the log file storing the user activities can be deleted or manipulated by the admin. In case of lack of bandwidth to support multiple user at the same time, or in case an empty database is tried to be read or a full database is tried to be written, the denial of service might happen on it.

The processes that exist within the trust boundary need not be considered for threats since they are not open for attackers to be exploited. So the processes like login process, role and access definition/ modification, work flow definition can be ignored for this purpose. The rest processes under consideration are subject to all six kinds of attacks according to STRIDE model. The admin authentication process is open to spoofing like url spoofing or phishing. But the process register or modify SCM is not subject to such attack since it is the outcome of successful login. But unless the session parameters are properly set and certificates are properly verified, a spoofing attack also cannot be made out of focus. The credential and sensitive information from the admin may be taken away by this type of attack. It is really hard in the present scenario to read process memory and change it according to our need, so it can be said that the information disclosure threats on the processes are not to be considered. In worst case if the operating system running the software can be compromised, the information disclosure may happen. But however by the buffer overrun attack, the process memory can be tampered which may lead to process tampering. But since the system being developed is using Java development platform, buffer overrun attack is a kind of impossible thing to do.(tampering in process is a case if it is spoofed) There will be little chance of process repudiation in the processes. If the processes are considered for denial of service, it is evident that because a very limited number of users get to register to the system, so the process is not going to have any DOS threat, as per requirement. (But if the number of users increases beyond the pre-estimated number without proper modification in the process parameters,

resource management and bandwidth management, there may be a chance of this type of threat). Since the whole system is being written in Java, elevation of privileges possibility is little (because of lack of pointer use and array boundary checking).

The data flows that are not across the trust boundary are not to be considered as they are internal to the system. The three data flows: login request, scm specific information flow and assignRole() request flow are to be considered. As per STRIDE, the data flows are subject to TID type of attacks. Tampering in data flow is very much likely to happen in case of weak cryptographic integrity control or by replaying attacks or when duplicate and overlapped data are allowed or if there is a chance of man in the middle attack. In case of weak encryption decryption algorithms, lack of ACLs or any side channel analysis, confidential data may be revealed leading to information disclosure. There is no chance of denial of service on the data flows.

The attacks possible on the Admin module is shown briefly in table **??**.

Table 4.1: Threats to Admin module

	External Entity	Data flow	Database	Process
Spoofing	- IP Spoofing - Session Hijacking - Offline password attacks - Man in the middle attack - XSS	NA	NA	- DNS Spoofing - ARP poisoning - URL spoofing - Content spoofing - MITM
Tampering	NA	- Sniffing attack - Replay Attack - MITM	- SQL injection	NA
Repudiation	- Repudiation Attack - Log Injection - Web parameter tampering by MITM	NA	- Log file manipulation via SQL injection - Privilege to Admin of the Log files	NA
Information Disclosure	NA	- Side channel Analysis -Sniffing	- SQL Injection	NA
Denial of Service	NA	NA	- Empty DB tried to be read or full DB tried to be written - Forced browsing - Resource consumption attacks	- DOS attack - XSS, a link may redirect to another one leading DOS for actual link
Elevation of Privileges	NA	NA	NA	XSS

Data input module:

The customer data analyst is the only external entity here, hence it is liable to spoofing and repudiation assaults. By session fixation or session forecast attacker

can pick up the benefits of the him and upload the data of his own organization without authorization. Likewise if the message exchanged between the framework and the customer data analysis not legitimately encrypted and could be perused by the intruder he may launch man in the middle attack. Indeed in the less complex and most pessimistic scenario, the certifications of credentials may be stolen in the event that it is put away with him in a content document or it might be speculated by the attacker. Without the proper support of log, disarray may happen if the customer data analyst latter denies to have done some action, prompting repudiation.

There are two data bases in the module: user schema and temporary database. The user schema behaves the same as has been mentioned in the previous admin module the temporary database acts as a temporary buffer. The input buffer gets un-trusted information - its substance are not viewed as reliable. There is no real way to adjust the current substance of the database by sending data remotely. Consequently, the danger of tampering does not make a difference here. There is a little danger of information disclosure- it is conceivable that a side channel attack may disclose the limit and/or the capacity of the buffer. There is a substantially more critical danger of denial of service, since the data stay in the buffer until sent or dropped.

The processes which are not interacting data flows across the trust boundary are not to be considered. User authentication is the same as it has been describes in the admin module. On Raw data importing module, elevation of privileges and tampering are not possible. Information disclosure is possible only if the OS is compromised by the adversary. Denial of service is a possible threat since the uploading of huge amount of data uses heavy resource and time.

The data flows that are not across the trust boundary are not to be considered as they are internal to the system. The three data flows: login request, item and hierarchy definition flow and hierarchy and causal data definition flow are to be considered. Tampering in the data flows is very likely to happen in case of weak cryptographic integrity control or by replaying attacks or when duplicate

and overlapped data are allowed or if there is a chance of man in the middle attack. In case of weak encryption decryption algorithms, lack of ACLs or any side channel analysis, confidential data may be revealed leading to information disclosure. Denial of service is a case here because on one session huge amount of data flows over the traffic and process uploading the data is not doing it that much speedily.

The attacks possible on the Data Input module is shown briefly in table 4.2.

Table 4.2: Threats to Data Input Module

Threats	External Entity	Data flow	Database	Process
Spoofing	- IP Spoofing - Session Hijacking - Offline password attacks - Man in the middle attack - XSS	NA	NA	- DNS Spoofing - ARP poisoning - URL spoofing - Content spoofing - MITM
Tampering	NA	- Sniffing attack - Replay Attack - MITM	NA(for temp DB) - SQL injection for User schema	NA
Repudiation	- Repudiation Attack - Log Injection - Web parameter tampering by MITM	NA	- Log file manipulation via SQL injection - Privilege to Admin of the Log files	NA
Information Disclosure	NA	- Side channel Analysis -Sniffing	- SQL Injection	NA
Denial of Service	NA	NA	- full DB tried to be written, empty user DB may be tried to be read - Forced browsing - Resource consumption attacks - Huge Data stays in DB until sent in temp db, better chance of DOS	- By spoofing a user, -DOS attack - XSS, a link may redirect to another one leading DOS for actual link
Elevation of Privileges	NA	NA	NA	XSS

Data setup module:

Here the two external entities SCM data manager and Customer data manager behave the same way as in case of the previous two module external entities. They are susceptible to spoofing and repudiation as in the above two cases. Session fixation, session prediction, man in the middle attack are the measure IP spoofing attacks and IP spoofing may cause denial of service in both the cases also, if the number of simultaneous users exceed beyond capacity.

In this module, there is interaction with all four databases. The user schema database works as defined in the admin module. The customer specific data is retrieved from the temporary database which is done by an internal process, hence

it is not liable to any kind of threats after the upload process is over. It is not also subjected to denial of service since data is read here, not written. The staging database contains temporary files so no need of log trails hence no repudiation. More over there is no tampering or information disclosure since it does not contain any proper data, rather it contains only intermediate data. Denial of service is possible in only case when the database memory is very less and it becomes full while it is still tried to be written. The main database contains the filtered information which is uploaded after all processes. Hence it is liable to denial of service because of huge data interaction and simultaneous user upload may also be tried to it. No tampering is possible since process uploading is the internal one and no user interaction during this. but this database is accessed by the customer planning manager for review purpose; so on spoofing him, attacker can gain full access to the main database and in that case, information disclosure becomes a serious issue. In overall case, unencrypted data in database is liable to information disclosure and tampering once the attacker gets access to it.

There are 8 processes in this module involved in the process of interacting through the trust boundary. The user authentication one behaves just like it did in the previous two modules. Spoofing of the rest of the processes is very much possible to gain access to the different metrics, rules, definitions being transmitted. Information disclosure of processes is unlikely unless the attacker gains access to the OS. Repudiation also does not need special mention. Tampering may happen if buffer overflow works which is a rare case in the development platform. Denial of service is unlikely in every cases except the review process between the customer planning manager and the main database where the traffic may be kept busy by any one transactions and hence resulting a system slow down.

The data flows that are not across the trust boundary are not to be considered as they are internal to the system. There are 10 data flows are to be considered. Tampering in the data flows is very likely to happen in case of weak cryptographic integrity control or by replaying attacks or when duplicate and overlapped data are allowed or if there is a chance of man in the middle attack. In case of weak

encryption decryption algorithms, lack of ACLs or any side channel analysis, confidential data can be revealed leading to information disclosure. Denial of service is a case in the request for review flow from the customer planning manager to the database here since on one session huge amount of data flows over the traffic and process uploading the data is not doing it that much speedily.

The attacks possible on the Data Setup module is shown briefly in table 4.3.

Table 4.3: Threats to data setup module

Threats	External Entity	Data flow	Database	Process
Spoofing	- IP Spoofing - Session Hijacking - Offline password attacks - Man in the middle attack - XSS	NA	NA	- DNS Spoofing - ARP poisoning - URL spoofing - Content spoofing - MITM
Tampering	NA	- Sniffing attack - Replay Attack - MITM	NA(for temp DB and staging DB) - SQL injection for User schema	NA
Repudiation	- Repudiation Attack - Log Injection - Web parameter tampering by MITM	NA	- Log file manipulation via SQL injection - Privilege to Admin of the Log files -NA for staging DB	NA
Information Disclosure	NA	- Side channel Analysis -Sniffing	- SQL Injection -NA for staging DB	NA
Denial of Service	NA	NA	- full DB tried to be written, empty user DB may be tried to be read - Forced browsing - Resource consumption attacks - Huge Data stays in DB until sent in temp db, better chance of DOS -NA for staging DB	- By spoofing a user, - DOS attack - XSS, a link may redirect to another one leading DOS for actual link
Elevation of Privileges	NA	NA	NA	XSS

Structural analysis module:

Here two external entities are present: system and SCM planning manager. Since the system is the automated one, no question arises of spoofing attack or repudiation attack on it. The SCM planning manager is subject to spoofing and repudiation attacks. By session fixation or session prediction adversary may gain the privileges of him and ill-function his privileges to review and change the reports. Also if the message transferred between the system and him is not properly encrypted and can be read by the adversary, he may launch man in the middle attack. Without the appropriate maintenance of log, confusion might happen if the admin latter denies to have done some activity, leading to repudiation.

Here the database involved is main database and a small LDAP or active directory is used for storing the output reports. The main database is subject to attacks as has been stated in the data setup module with the interaction of SCM planning manager. The report store is subject to tampering and information disclosure once the SCM planning manager is spoofed. Repudiation threat does not happen though there is a maintenance of log because there is no capability of the user to tamper it. But in worst case, the admin may be given the privilege to the log and he tampers or delete it. Denial of service is not a considerable threat here for report store.

Processes here are completely internal since they are controlled by the system entity. So no threat to the processes crossing the trust boundary. Only threat can happen to the review process which has already been discussed in the data setup module.

Data flows except the request for review and respective response are secure. Tampering in the data flows is very likely to happen in case of weak cryptographic integrity control or by replaying attacks or when duplicate and overlapped data are allowed or if there is a chance of man in the middle attack. In case of weak encryption decryption algorithms, lack of ACLs or any side channel analysis, confidential data may be revealed leading to information disclosure. However, in this case there is no denial of service since not much amount of data interaction happens with the report store.

The attacks possible on the Data Setup module is shown briefly in table 4.3.

Output unit:

no new threats to this unit, if the threats to the other modules are resolved properly. However, proper validation of the output unit is required here to ensure that the output is shown to the proper SCM data analyst.

Tables below show briefly all the attacks on all the modules briefly in a tabular manner.

Taking all module threats all together, the number is found out as 10 spoofing attacks , 21 tampering attacks, 9 repudiation attacks,21 cases of information

Table 4.4: Threats to Structural analysis Module

Threats	External Entity	Data flow	Database	Process
Spoofing	-NA for system - IP Spoofing - Session Hijacking - Offline password attacks - Man in the middle attack - XSS	NA	NA	- DNS Spoofing - ARP poisoning - URL spoofing - Content spoofing - MITM
Tampering	NA	- Sniffing attack - Replay Attack - MITM	- SQL injection for User schema and Main DB	NA
Repudiation	-NA for system - Repudiation Attack - Log Injection - Web parameter tampering by MITM	NA	- Log file manipulation via SQL injection - Privilege to Admin of the Log files	NA
Information Disclosure	NA	- Side channel Analysis -Sniffing	- SQL Injection	NA
Denial of Service	NA	NA	- full DB tried to be written, empty user DB may be tried to be read - Forced browsing - Resource consumption attacks - Huge Data stays in DB until sent in main db, better chance of DOS	- By spoofing a user, -DOS attack - XSS, a link may redirect to another one leading DOS for actual link
Elevation of Privileges	NA	NA	NA	XSS

disclosure, 8 cases of denial of service and 10 cases of elevation of privileges have resulted. The next step is the suggestion for remediation.

In this phase the suggestion for remediation is given in the design document.

To stop the system from spoofing attack, a strong authentication technique has to be implemented at all interfaces with the external entities. The credentials should be random and arbitrary. The secrets should not store the secrets on the server (for example, public key.) If it is storing shared secrets, hashing or encryption has to be applied to them with appropriate salt. Credentials at the client should be remembered , not stored in machine as it may get stolen. If storing them is required, encryption should be applied to them. Existing and reliable key distribution system should be relied upon. Strong cryptography should be there for the transmission of data along the network. Protocols for updating credentials are the common attack vectors. For example "forgot my password" function can be used as an attack vector. A proper standard implementation through secure coding has to be implemented. Use of CAPTCHA should be there to prevent brute-forcing attack. Sometimes new attack techniques make a secure application vulnerable and update of the software is called for to resolve the security problem.

The old clients who are still using the previous code should be updated to the new one and the previous less secure methods should not be enabled by default leaving them vulnerable. This is feature is called backward compatible mode and this scope should be there in the secure coding mechanism.

To stop the system from repudiation attack, the user activity should be logged. Standard digital signature scheme should be introduced. An anti-replay technique and a strong integrity technique have to be followed. It is because data flows without sequence numbers or timestamps may be captured and they may be replayed by the adversary. So the dataflow has to be defended by the hashing or MAC or digital signature technique and time stamp or counters have to be followed. Sufficient space should be there for the activity log so that it does not run out of space. The process should be made to run at a lower privilege level not to tamper with the logs. Logs in database should be made to contain enough data to analyze things after even the deletion of the logs. So they stay well protected. Maintenance of ACL along with protection of Audit logs to be done.

To stop the system from tampering attack, Cryptographic integrity control for the data in network has to be done. An anti-replay technique and a strong integrity technique has to be followed. To prevent the man in the middle attack, the end points should be authenticated to each other before the start of the session. So a key persistence algorithm (a la SSH) or a PKI (public key infrastructure) or Kerberos has to be implemented Standard protocol like SSL has to be adopted for a strong message integrity system. The keys have to be kept secret for a strong channel integrity system. Integrity control and careful input validation has to be done. Permissions have to be set properly by giving least required permissions to each user, even to Admin. The program has to be ensured to be the only thing to access the database and the users or any other system can have the API only to interact with the system and there by the database. Error should be returned in case database is full. This case arises when either the data is discarded when full or the data storage wraps to its start when it is full. Again, the error should be a generic one and it should not reflect the details of the internal process.

To stop the system from information disclosure, The data in the database as well as flowing across the system should be considered for encryption and the access control list should be introduced on the system to access it so that it is not going to be used by anonymous users, in case any problem occurs. The key exchange or key validation should be performed out of band to ensure that all end points are mutually authenticated. Encryption should be done to the messages as it crosses the network. A strong channel confidentiality system and message confidentiality system should be ensured. Constant time approach should be applied to encryptions to increase the chance of un-ambiguity in the encrypted message to prevent side channel attacks. In case of an undo or recovery, the files should be cleaned explicitly and its rarely a performance issue.

To stop the system from Denial of service, permissions have to be properly set on database. All names should be hard to predict and well protected. The file system should not be shared and the registry access across different trusted parties should not be shared. The app should deal with an unavailable data store to make fool to the attacker. Log for that false data store should be kept also. Bandwidth calculation and then allocation has to be done for the system data flow and database accesses. Sufficient amount of memory should be available for the whole operation of the system.

To stop the system from Elevation of privileges, Careful validation of all user input for appropriate purposes otherwise it can lead to buffer overflow. To validate all inputs, a white list to be maintained.

After this phase, threat prioritization and risk analysis is done on the system. In the given software, it has been found that there are 23 high risk threats, 13 are medium risk attacks and rest 43 are low risk threats present.

In the same way, threat modeling to the TCS Intellectual Property Asset Registry (TIPAR) system has also been done.

4.2 Threat modeling to TCS Intellectual Property Asset Registry (TIPAR) system

The threat modeling to the system is done in the same way as it has been described in the previous system. So without going into the details of the process of threat modeling, only the overview has been described. For conciseness purpose, they are not depicted in a detailed manner, but the system overview has been given out of which the omitted parts can be derived just like the previous system.

TIPAR system is a collection of all the TCS Intellectual properties that are owned by the TCS employees as well as the other organizations. The employees are open to search all the assets on the system and if there exists an asset demanded, he can send request to the owner of the system to provide his asset to him. The asset is evaluated and monetized according to the evaluation and number of uses. The money is paid by the employee and asset is used by him. The employee can also register his own asset by requesting the operating unit single point of contact person(IO SPOC) to register in the system. The asset is verified. Its valuation and monetization as well, is verified by the verifier. After successful verification, asset gets registered else its sent back to the employee for further modification. So this is the business objective of the system.

The security objectives can be the same as previous: satisfaction of all security properties that are confidentiality, integrity, availability, accountability etc.

The system overview diagram ie the context diagram is shown in Figure4.8.

The different tasks are again divided into modules which is shown in the decomposed level 1 data flow diagram (Figure4.9).

The threat identification and the following phases are similar to those in the previous scientific forecasting system.

The threatened elements in the decomposed diagrams in both the applications are shown in Table 4.5. The table depicts the number of threatened elements combining al modules together for a system for six different classes of STRIDE threats.

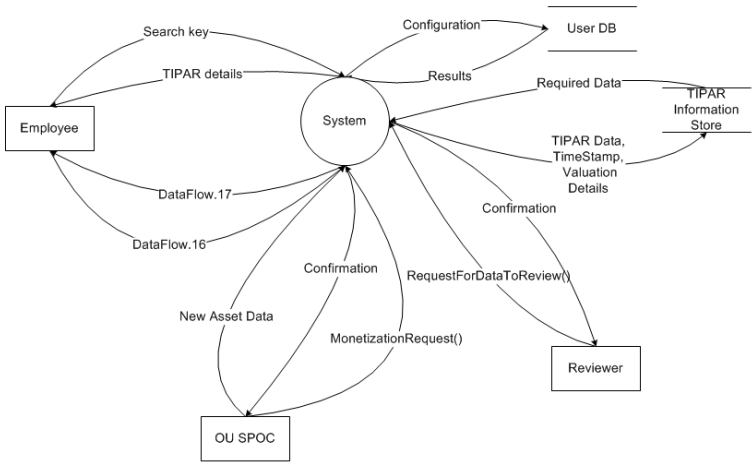

Figure 4.8: Context Diagram of TIPAR system

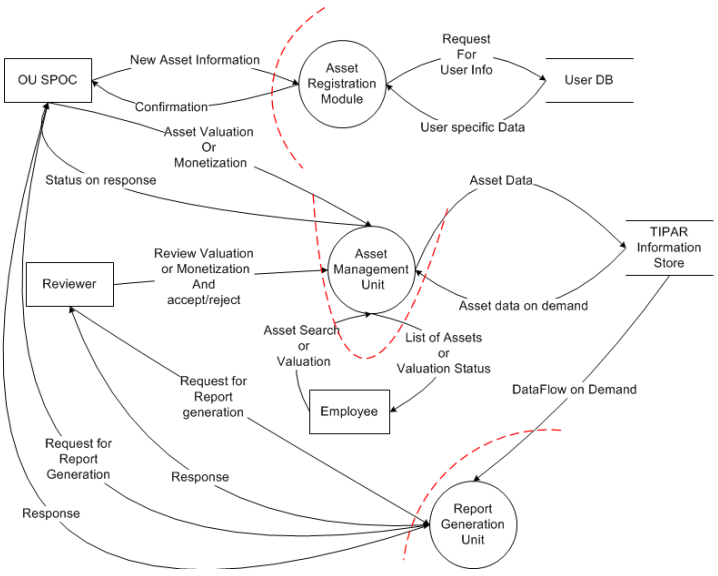

Figure 4.9: Level 1 DFD of TIPAR system

Table 4.5: No. of threatened elements in two industrial projects

Threat	No. of threatened elements in Scientific Forecasting System	No. of threatened elements in TIPAR System
Spoofing	10	6
Tampering	21	17
Repudiation	9	5
Information Disclosure	21	17
Denial of Service	8	5
Elevation of Privileges	10	12

Chapter 5

Modification to the existing Hybrid Approach

In this approach Data flow diagrams instead of Misuse case diagrams to show the threats has been used in the hybrid approach of threat modeling. Hence the second and third phase of the hybrid threat modeling process, the functional and security requirement identification phase have been modified. The modifications to these phases result in a data flow diagram describing the information flow and threats to each information and entity of the systems respectively. The motivation behind doing this is described as follows:

5.1 Motivations behind the modification

• In the existing approach, the use case diagram and misuse case diagrams are used to do so. This diagram works well, but it is not appropriate to use them as the primary way to find out and document business process requirements. A Use Case diagram shows a single activity, but doesnt show an entire process flow or any information flow. It is not good for a business process analysis if the graphical representation of information flow that flows into, within, and out of the business is not shown.

• For a misuse case diagram, a text record also has to be maintained showing the details of each function in detail, which has been represented through misuse case template in the existing hybrid approach. It adds an extra overhead to the process of threat modeling after the brain storming and drawing the misuse case

diagram. In contrast, using the data flow diagrams, only on the functional behavior model, the threat types can be added, description and mitigation suggestion of it without use of any extra diagram or templates can be depicted.

• In the existing hybrid threat modeling approach, there is no report generation module for the final threat model. Generally in industries, for the development process lifecycle of any application, the technical persons, whether they are security aware persons or not, refer to reports which describe the threat profile and mitigation suggestions in easier language that can be understood by all. Without this report, its too hard to interpret everything unless well aware of everything. In the existing approach, misuse case diagrams, misuse case templates and threat trees together have to be gone through to interpret the threat profile. In contrast to the clumsy technique, better to prepare a threat report that describes everything, that will be easier for developers to prepare and easier for readers to understand.

• The threat representation and prioritization of threats in the existing approach is done using attack tree. In the proposed approach, this concept may be still relied upon, though the threat representation through attack tree is not needed any more after the threat report. The threat report is itself a threat representation. Another report generation feature can be added to the system which shows the threat priority to the threats.

• The existing approach claims that it follow the STRIDE methodology to derive the threat profile in the Misuse case diagram. However, there is no verification technique implemented for it since it is purely unsystematic and thought-dependent with no traces of STRIDE in the benchmark implementing it (Suraksha). It would be better if the STRIDE specification can be shown while defining the threat profile, which is done in the proposed approach.

5.2 Modifying the existing tool

The implementation of the proposed approach has been done on the framework of Suraksha, the security workbench that has been developed to support the ex-

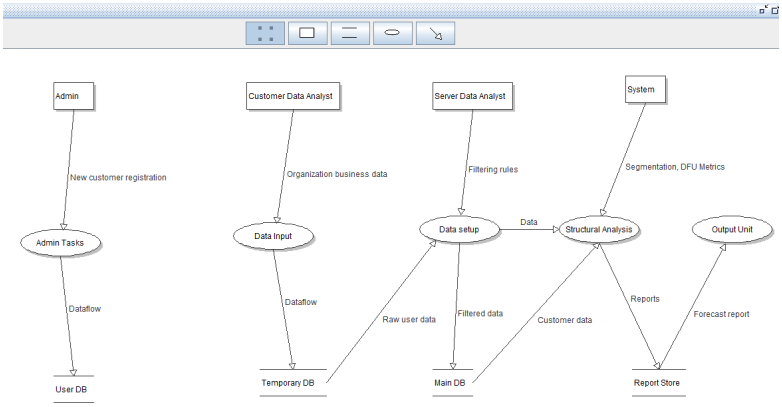

Figure 5.1: DFD implementation in Suraksha Tool

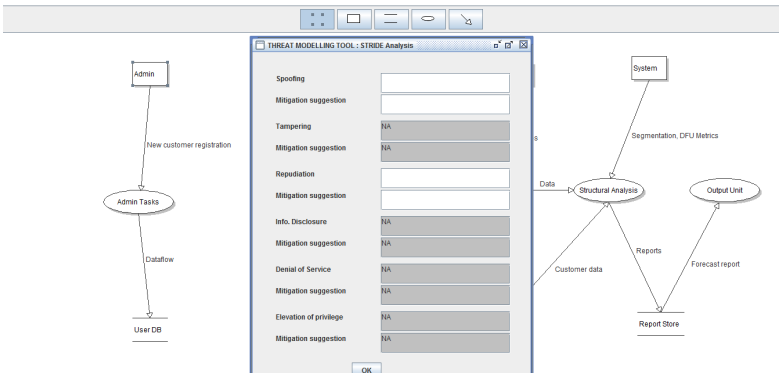

Figure 5.2: STRIDE for different elements of DFD in Suraksha

isting hybrid approach. The snapshots of the implementations are shown as the following diagrams. Figure5.1 shows the data flow diagram implementation on the Suraksha framework and Level 1 DFD of Scientific forecasting system drawn upon it. Figure5.2 shows STRIDE implementation on individual elements of the DFD as explained earlier in the section. Figure5.3 shows the modified Suraksha toolbar menu indicating the extra addition of the menu for report generation after the complete dfd and the elements' corresponding STRIDE threats and mitigation suggestions have been mentioned. Figure5.4 shows a demo of the report generated after the complete threat modeling process using the proposed approach.

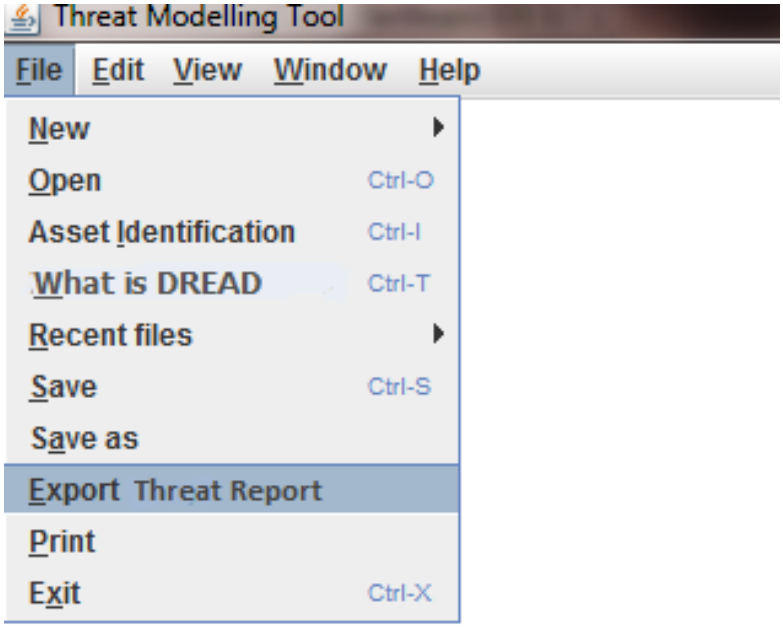

Figure 5.3: Report Generation Capability Introduced in Suraksha

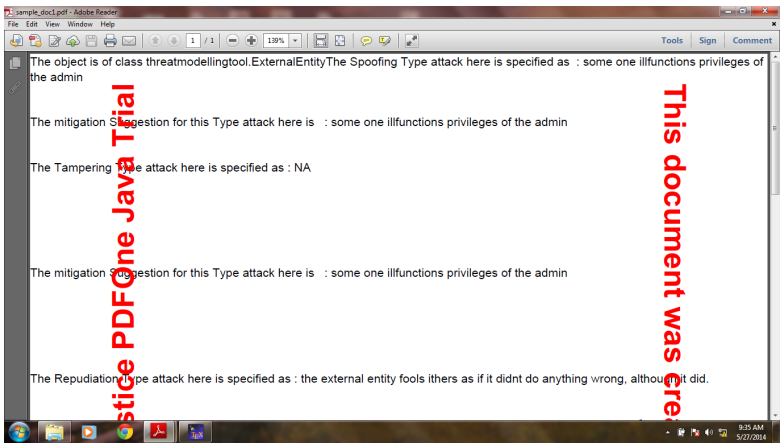

Figure 5.4: Report generated after threat modeling

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Threat modeling is applied not only applied to web applications but also to embedded systems, cloud applications, wireless sensor networks, network tools etc for threat evaluation and risk analysis along with mitigation suggestions to them. Threat modeling for a application takes a lot of brainstorming sessions to collect all information of the assets, trust boundaries and threat profiles possible on the assets. The approach of Microsoft is followed by most of the application developing companies and is the most acceptable one. Along with threat evaluation, it takes care of business aspects of a software in a stipulated time period. This is a software centric approach. Currently software centric approach dominates over the other two. However it is beneficial to use the combined approach. Whenever it comes to industries, a hybrid approach with a report generation capability is hoped to get preferred.

In the thesis the detailed step by step process of threat modeling is explained and illustrated via an example. The threat modeling of two industrial applications has been done and one has been explained in greater details. The existing hybrid approach for threat modeling has been explained step by step. The proposed work for some improvements in it has been mentioned with reason and the implementation of the proposed scheme on the hybrid approach supporting tool has been implemented. The works have been carried out in utmost care and any further modification is cheerfully appreciated.

6.2 Future scope of work

The main drawback of the threat modeling tools has been the lack of automation. The tools can not directly detect the threats rather classify them into some threat class statically. This makes the tools work kind of insignificant in the threat modeling process. so it will be a great contribution of researchers to add the automation feature to the system whenever possible to the system. Libraries containing security modules or algorithms should be attached to the tools, as an afterthought, for the scalability of the threats in future.

Bibliography

[1] D. Xu and K. E. Nygard, "Threat-driven modeling and verification of secure software using aspect-oriented petri nets," *Software Engineering, IEEE Transactions on*, vol. 32, no. 4, pp. 265–278, 2006.

[2] F. S. Labs, "Threat report h2 2013[Online]." `http://www.f-secure.com/en/web/labs_global/whitepapers/reports`, April 2014. Last Accessed: 27-03-2014.

[3] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[4] M. Corporation and iSEC Partners, "Microsoft sdl: Retun on investment[Online]." `http://www.microsoft.de/sdl`. Last Accessed:1 March 2014.

[5] F. Swiderski and W. Snyder, *Threat modeling.* " O'Reilly Media, Inc.", 2009.

[6] F. Swiderski and W. Snyder, "Threat modeling, 2004."

[7] P. Torr, "Demystifying the threat modeling process," *Security & Privacy, IEEE*, vol. 3, no. 5, pp. 66–70, 2005.

[8] "Why dfds? when swim lanes are not enough a comparison of process modeling techniques." Web Site: http://www.advstr.com.

[9] A. K. Talukder, V. K. Maurya, B. G. Santhosh, E. Jangam, S. V. Muni, K. Jevitha, S. Saurabh, and A. R. Pais, "Security-aware software development life cycle (sasdlc)-processes and tools," in *Wireless and Optical Com-*

munications Networks, 2009. WOCN'09. IFIP International Conference on, pp. 1–5, IEEE, 2009.

[10] S. Lipner, "The trustworthy computing security development lifecycle," in *Computer Security Applications Conference, 2004. 20th Annual*, pp. 2–13, IEEE, 2004.

[11] Microsoft, "Simplified implementation of the microsoft sdl," 4th November 2010. For the latest information, please see http://www.microsoft.com/sdl.

[12] Wikipedia, "Microsoft security development lifecycle[Online]." `http://en.wikipedia.org/wiki/Microsoft_Security_Development_Lifecycle`. Last Accessed: 10-05-2014.

[13] S. F. Burns, "Threat modeling: A process to ensure application security," *GIAC Security Essentials Certification (GSEC) Practical Assignment*, 2005.

[14] S. W. Ambler, "Introduction to security threat modeling," 2010.

[15] J. Steven, "Threat modeling-perhaps it's time," *Security & Privacy, IEEE*, vol. 8, no. 3, pp. 83–86, 2010.

[16] D. Dhillon, "Developer-driven threat modeling: Lessons learned in the trenches," *IEEE Security and Privacy*, vol. 9, no. 4, pp. 41–47, 2011.

[17] C. Mockel and A. E. Abdallah, "Threat modeling approaches and tools for securing architectural designs of an e-banking application," in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pp. 149–154, IEEE, 2010.

[18] E. A. Oladimeji, S. Supakkul, and L. Chung, "Security threat modeling and analysis: a goal-oriented approach," in *Proc. of the 10th IASTED International Conference on Software Engineering and Applications (SEA 2006)*, pp. 13–15, Citeseer, 2006.

[19] M. Nickolova and E. Nickolov, "Threat model for user security in e-learning systems," *International Journal Information Technologies and Knowledge*, vol. 1, pp. 341–347, 2007.

[20] R. Klöti, "Openflow: A security analysis," in *8th Workshop on Secure Network Protocols (NPSec 2013), Göttingen, Germany*, 2013.

[21] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Symposium on requirements engineering for information security (SREIS)*, 2005.

[22] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat modeling-uncover security design flaws using the stride approach," *MSDN Magazine-Louisville*, pp. 68–75, 2006.

[23] Microsoft, "Introduction to threat modeling - microsoft." Available at: http://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Introduction˙to˙Threat˙Modeling.ppsx.

[24] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Uncover security design flaws using the stride approach" msdn. microsoft. com, nov. 2006."

[25] J. Jesan, "Threat modeling web-applications using stride average model," in *Computer Security Conference*, 2008.

[26] M. Abi-Antoun, D. Wang, and P. Torr, "Checking threat modeling data flow diagrams for implementation conformance and security," in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pp. 393–396, ACM, 2007.

[27] O. Guide, "Threat risk modeling," *A Guide to Building Secure Web Applications and Web Services,July 2005*, p. 34.

[28] K. R. M. Rao and D. Pant, "A threat risk modeling framework for geospatial weather information system (gwis): a dread based study," *international Journal of Advanced Computer Science and Applications*, vol. 1, no. 3, 2010.

[29] B. Schneier, "Attack trees," *Dr. Dobbs journal*, vol. 24, no. 12, pp. 21–29, 1999.

[30] I. Morikawa and Y. Yamaoka, "Threat tree templates to ease difficulties in threat modeling," in *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pp. 673–678, IEEE, 2011.

[31] V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 124–131, 2008.

[32] S. Chun Wei (Johnny), "Misuse cases and abuse cases in eliciting security requirements."

[33] I. F. Alexander, "Modelling the interplay of conflicting goals with use and misuse cases.," in *GBPM*, 2002.

[34] G. Sindre and A. L. Opdahl, "Templates for misuse case description," in *Proceedings of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), Switzerland*, Citeseer, 2001.

[35] G. Santhosh Babu, V. K. Maurya, E. Jangam, V. Muni Sekhar, A. K. Talukder, and A. R. Pais, "Suraksha: A security designers workbench," *Proc., Hack. in 2009*, pp. 59–66, 2009.