Threats Indentification in Web Application

Dr Gayatri Devi
Professor, Department of Computer Science and Engineering, ABIT,Sec-1,CDA, Cuttack, Odisha, India.

Rajeeb Sankar Bal
Senior Lecturer, Department of Computer Science and Engineering, ABIT,Sec-1,CDA, Cuttack, Odisha, India.

Pragyan priyadarsini Sahoo
Student, M.Tech(CSE), Department of Computer Science and Engineering, ABIT,Sec-1,CDA,Cuttack,Odisha, India.

Abstract –. Now a day's man wants to live in an intelligent and smart environment, an environment that would make life more easy and comfortable, enhancing the quality of his living, with various intelligent automation devices and services. The Hazards is applied not only applied to web applications but also to embedded systems, cloud applications, wireless sensor networks, network tools etc for Hazards evaluation and risk analysis along with mitigation suggestions to them. Hazards for a application takes a lot of brainstorming sessions to collect all information of the assets, trust boundaries and Hazards pro les 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 Hazards evaluation, it takes care of business aspects of 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.

Index Terms –Operating System (OS), Software Development Life Cycle (SDLC), Software System (SS), Data Flow Diagrams (DFD), Final Security Review (FSR), Supply chain management (SCM).

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, data store, 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. Reliance on network security alone or installation of firewall is not sufficient as it does not address the logic errors, flaws in architecture of SS, flaws in operating system and its resource limitations or the design level problems. As it started, on 2nd 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, installation of firewall with a 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. So on the basis of the last two or three decade's security trend, innovative Threats 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, business continuation, and corporate reputation along with legal and regulatory perspectives properly. Risk management is a major goal in business applications, i.e. 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 ne-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 development life cycle to be considered along side of SDLC. In a 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 functions that the SS 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 requirements which includes all other aspects than the functional ones like assessment of cost, platform compatibility, monetary and decisions etc. So adding security requirements to the non-functional class is ne as security requirements do not come under what a SS is required to Rather, security requirements de ne the behavior that the system should have when Introduction 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 aw 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 requirements are considered as functional requirements. In fact there are several benefits if security is considered as functional requirement and it's considered in the SDLC 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 counter measure against each threats right from the earlier stage of SDLC. 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 pre-venting 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 ne 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 i.e. 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 i.e. implementation phase starts 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 in-corporation. Hence at the design phase the security aspects are best added which is implemented in the implementation phase. Next comes the cost e effectiveness consideration which has also to be done in the design phase in which the only Necessary security implementation in the software is done leaving the not-much-needed parts that may unnecessarily consume cost and time. Threats is the process of design level security consideration (consideration includes identification, prioritization and mitigation) and to cost effectively do it, risk-based Threats is considered. Before the introduction of Threats, the exact meaning of Threats, vulnerabilities, exploitations, attacks and difference between them should be understood [1].

1.1. Basic terminologies

Threats: 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 Threats. 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 over flow a buffer in software running on the target, which is also known as buffer over flows. 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].

1.2. Threat

A Threat is a procedure for optimizing network security by identifying objectives and vulnerabilities, and then defining countermeasures to prevent or mitigate the effects of threat to the system. It has emerged as an independent and comprehensive methodology. Many researchers have been taken place for the advancement of this area. A threat 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. Hence again it can be defined as a structured and formal approach of presenting, assessing and documenting security risks of a particular software. Threats cannot explicitly be considered as mathematical science and hence with the freedom and exibility even a non-security expert can exercise it with a provided convenient framework and support (though not with full e efficiency). As discussed before, it's better to add security suggestions right in or before the design phase of the SDLC. The same is followed by threat 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 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 [1].

1.3. Different approaches of Threat

- Asset-Centric: Asset-centric Threats involves starts with dentifying critical assets. As-sets 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.
- Attacker-Centric: Attacker-centric threat 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.
- Software-Centric: Software-centric threat, also termed as 'design-centric,' system-centric', or 'architecture-centric', starts with the design of the system. It involves application decomposition and pro ling, identifying threat 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 may start with 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 threat are to be assessed. 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 [2].

2. SECURITY DEVELOPMENT LIFE CYCLE

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 Threats, that has been discussed previously, is synonymous to security development life cycle. The proper security development life cycle was described by Lipner and Steve in the paper 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. 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 describes its own approach. 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.

2.1. 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 e 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 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 divided into five phases roughly:

- Training, policy, and organizational capabilities
- Requirements and design
- Implementation
- Verification
- Release and response

The first stage may be excluded from the stage schedule if it is considered to be a onetime 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. 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 shown in figure.

Figure 1. SDL Optimization Model with capability and maturity levels.

2.2. 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, and the ongoing trends in security and management of privacy. This training can be continued on a scheduled manner in a year in which the technical persons (design persons, developers, testers etc.) are mandatory to attend. The training areas include

- Face area, basic understanding of defense, least privilege adherence, the security adopted by default etc.
- Threats: It includes topics like threats overview, designing of
- A threats model, implementation constraints sticking to the threats model etc.
- Secure coding: It includes understanding of buffer over flow(in
- C, C++), arithmetic errors(in C, C++), XSS, SQL Injection, weak cryptography etc
- Security testing: it includes understanding of the difference
- between functional testing and security testing, risk assessment, methods of security testing etc
- 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.
- Miscellaneous: topics like advance security architecture and
- Design, depend-able UI design, detailed studies of security flaws and vulnerabilities, implementation of manual threat mitigation etc.

2.3. Requirements and design

- Security requirement

  For a secure SS 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 requirement by the customer lets the organization identify important milestones, deliverables and 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 a 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 personnel who might manipulate project specific clarifications and more appropriate security requirements. With all the clarifications and improvements; 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 information in such assessments are like find out the modules of the project that need Threats models beforehand of release, modules of project demanding security design re-views 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 (i.e. firewall, IPSec, kerberos or SSl etc). So there is a chance that implementation of security features is 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 specification describes the complete and accurate secure implementation of all functionality provided by a given feature or functions or in other words secures 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.

- Threats

This is what the whole thesis is about. It allows development teams to analyze, document and mitigation suggestion of the potential threat 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.

2.4. 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.

2.5. 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.

- Threats 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.

2.6. Release

- Incident Response Plan

In worst, it might be the case that programs with absolutely no known vulner-abilities 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.

- FSR

  Prior to release, it is the detailed assessment of all the security activities performed in an application system by the security advisor with the assistance from the technical development personnel's and the security and privacy team personnel's. The FSR generally includes an assessment of the Threats models, tool output, input and output validations, exception requests, performance issues against the previously standardized quality gates. A FSR may be considered to be passed if the issues are mitigated properly. Passed FSR with exceptions if all the security and privacy issues identified by it are 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. Its 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 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 [2].

Figure 2 The Microsoft Security Development Lifecycle-Simplified

3. RELATED WORK

Threat is a structured process of identifying and documenting the vulnerabilities to threat 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 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 process continues.

3.1. The process of Threat

The process of threat 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 threat and mostly it is open to anonymous users to operate on. It demands careful security concern and resource managements.

The detailed process of threat 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 process, a brief introduction should be given of a data flow diagram that is going to be used in the approach of threat process proposed by Microsoft.

- DFD: 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, and 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. 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 owing through it. Figure 3 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 data flows among them are shown by labeled arrows. The threat process is a step by step process and is best described by Figure 4.

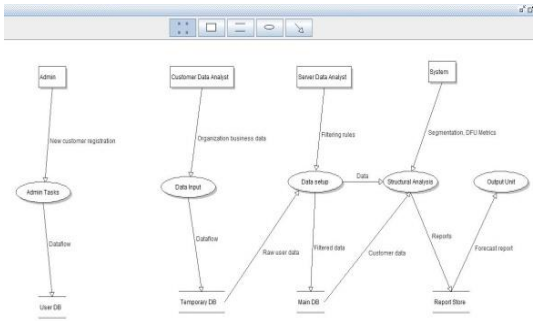

Figure 3 DFD example of an e-forecasting system.

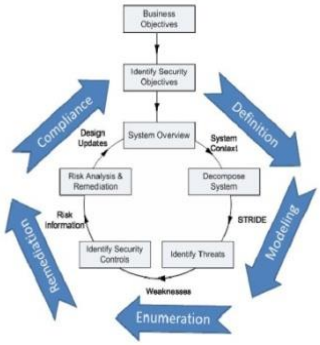

Figure 4 Threats Step by Step process.

The stages of threat process are shown in Figure 4 and are starting from business objective, identification of security objectives, system overview, and 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 [3].

4. THREAT IN LIVE WEB APPLICATIONS

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

4.1. Threat of scientific forecasting system

The threats of the scientific forecasting system have been explained in de-tail. The software is 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 Figure 5.

- 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 has the capacity in hundreds of Terra bytes. The customer sends business data and request to the system and gets his forecasted report back from 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 brie y stated as

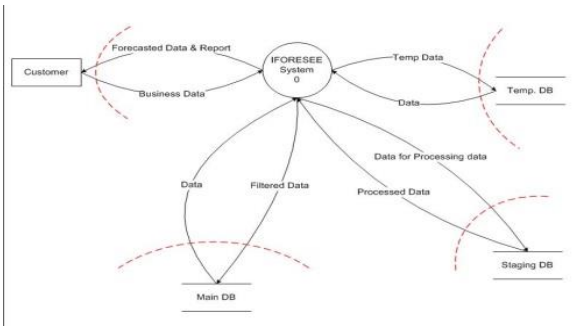

Figure 5 Context Diagram of Scientific forecasting system

- 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)
- 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)
- The system should provide uninterrupted service to the registered users.(satisfaction of Availability property)
- Identity of the user should be established (preferably by session parameters) before allowing access to the system. (Satisfaction of Authentication property)
- No other SCM should be able to see the confidential business data neither the output of other SCMs. (satisfaction of Authorization property)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 planning manager and for all the transaction histories. (satisfaction of Accountability property).

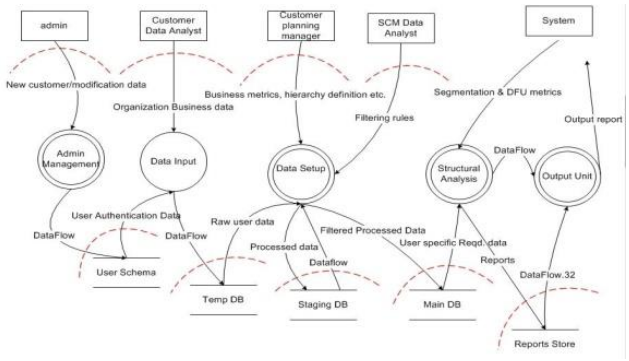

Figure 6 Level 1 DFD of Scientific forecasting

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 Figure 4.2. This diagram satisfies the system overview identification of Threats 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 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 preprocesses, filters, the data input by the customer data analyst 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 [4]

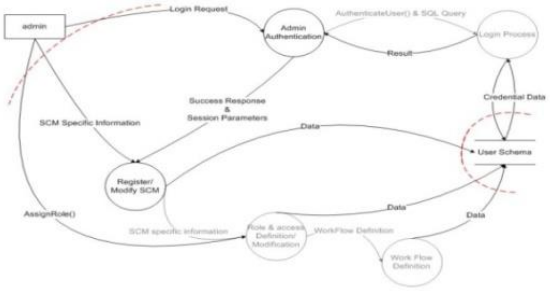

Figure 7 Admin Module

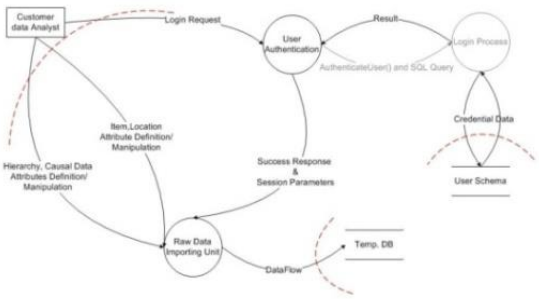

Figure 8 Data Input module own above.

In this approach Data flow diagrams instead of Misuse case diagrams to show the threats has been used in the hybrid approach of threats. Hence the second and third phase of the hybrid threats 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:

4.2 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 doesn't 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. In the existing hybrid threats approach, there is no report generation module for the final Threats 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 threats pro le and mitigation suggestions in easier language that can be understood by all. Without this report, it's too hard to interpret everything unless well aware of everything. In the existing approach, misuse case diagrams, misuse case templates and Threats trees together have to be gone through to interpret the Threats pro le. In contrast to the clumsy technique, better to prepare a threats report that describes everything, that will be easier for developers to prepare and easier for readers to understand. The threats 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 threats representation through attack tree is not needed any more after the Threats 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 follows the STRIDE methodology to derive the Threats 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 (Threat Report). It would be better if the STRIDE specification can be shown while defining the threats profile, which is done in the proposed approach.

4.3 Modifying the existing tool

The implementation of the proposed approach has been done on the framework of threats report, the security workbench that has been developed to support the Existing hybrid approach. The snapshots of the implementations are shown as the following diagrams. Figure 9 shows the data flow diagram implementation on the threats report framework and Level 1 DFD of Scientific forecasting system drawn upon it. Figure 10 shows STRIDE implementation on individual elements of the DFD as explained earlier in the section. Figure 11 shows the modified threats report 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. Figure 12 shows a demo of the report generated after the complete threats process using the proposed approach.

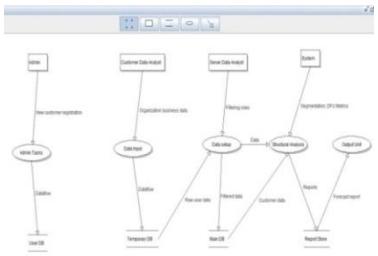

Figure 9 DFD implementation in threats report Tool

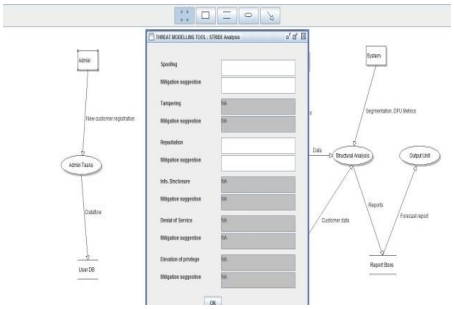

Figure 10 STRIDE for different elements of DFD in threats report

Figure 11 Report Generation Capability Introduced in Threat report.

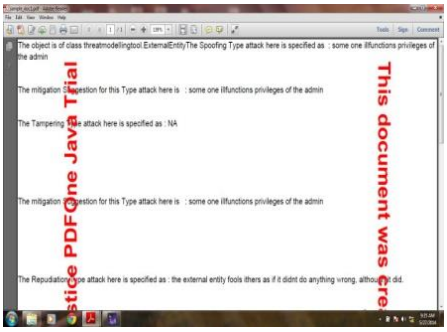

Figure 12 Report generated after Threat

5. CONCLUSION

Threats is applied not only applied to web applications but also to embedded systems, cloud applications, wireless sensor networks, network tools etc for threats evaluation and risk analysis along with mitigation suggestions to them. Threats

for a application takes a lot of brainstorming sessions to collect all information of the assets, trust boundaries and threats 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 threats evaluation, it takes care of business aspects of 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. The threats of two industrial applications have been done and one has been explained in greater details. The existing hybrid approach for threats 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.

REFERENCES

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

[2] 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 Communications Networks, 2009. WOCN'09. IFIP International Conference on,

[3] Wikipedia, \Microsoft security development lifecycle[Online]." http://en.wikipedia.org/wiki/Microsoft_Security_Development_Lifecycle.

[4] S. Myagmar, A. J. Lee, and W. Yurcik, \Threat modeling as a basis for security requirements," in Symposium on requirements engineering for infor-mation security (SREIS), 2005.