Letterkenny Institute of Technology

A thesis submitted in partial fulfilment of

the requirements for the Master of Science in Computing in

Enterprise Applications Development Letterkenny Institute of Technology

Threat Modelling for Legacy Enterprise Applications

Author:
Michael McGrath

Supervisor:
Ruth Lennon

Submitted to the Higher Education and Training Awards Council (HETAC)
August 2013

DECLARATION

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Master of Science in Computing in Enterprise Application Development, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution

Signature of Candidate: _____ Date: _____

DEDICATION

Do mo bhean chéile Karen agus ár bheirt mic, Oisín agus Micheál,

Míle buíochas ar son do chuid grá agus tacaíocht le linn mo chuid staidéir agus fá choinne mé a threorú agus a spreagadh i ngach rud a dheanam.

ACKNOWLEDGMENTS

I would like to thank my adviser, Ruth Lennon, for her guidance and support. Thanks to Niamh Hamill, Maura Schoumacker and Padraic Lynch for their valuable input. I would especially like to thank Company X for granting access to their enterprise system during this research and Lead Technical Developer Mr. N. for his contribution and encouragement. Further, I would like to thank my fellow students for sharing this journey. A special thanks to my family for their constant support. And thanks to my PlayStation III, for her patience and understanding during our 18 month separation.

ABSTRACT

Legacy enterprise applications provide unique challenges for software security personnel. The size and historical nature of these systems can result in vulnerabilities that do not have the appropriate countermeasures in place. Development teams that support these systems can be unaware of such security weaknesses until they have been exploited by an adversary. By successfully identifying threats, development teams can put in place the appropriate mitigations.

This research discusses the practice of Threat Modelling as a systematic approach to identifying security vulnerabilities in software systems. Although numerous works have been presented on the subject of Threat Modelling, very little has been published on the unique challenges faced with Threat Modelling legacy systems. This research presents different Threat Model methodologies and provides a comparison of leading practices suitable for the Threat Modelling of large scale systems. The comparison is based on both theoretical research and the practical application of two of the most popular Threat Models. This research then offers a Threat Model case study of a major component of a live commercial legacy enterprise application. An Irish based software company has provided access to an existing legacy system for the purpose of this project, the practical development of a Threat Model and a detailed analysis of the system.

ABREVIATIONS

AS/NZS	Australian/New Zealand Standard
BSIMM	Building Security in Maturity Model
CLASP	Comprehensive and Lightweight Application Security Process
CMMI	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model Integration for Develoeprs
CRUD	Create, Read, Write, Update, Delete
CVSS	Common Vulnerability Scoring System
DFD	Data Flow Diagram
DREAD	Damage Potential, Reproducibility, Exploitability, Affected users, Discoverability
EoP	Elevation of Privilege
HAZOP Analysis	Hazardous Operations Analysis
IT	Information Technology
IP	Internet Protocol
ISO	International Organization for Standardization
NIC	Network Interface Controller
OWASP	Open Web Application Security Project
P.A.S.T.A	Process for Attack Simulation and Threat Analysis
PTA	Practical Threat Analysis
S.B.S.R.S	Security Bulletin Severity Rating System
SA	Server Administrator
SAMM	Software Assurance Maturity Model
SDL	Security Development Lifecycle
SDLC	Software Development Lifecycle
SME	Small and Medium Enterprise
SSDL	Secure Software Development Lifecycle
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TAM	Threat Analysis and Modelling
WinPcap	Windows Packet Capture Library
WinSock	Windows Sockets API
XAMPP	Apache HTTP Server, MySQL database, PHP and Perl

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1.

INTRODUCTION

Over the last decade the global software industry has striven to develop new practices and methodologies which help software engineers build security into the development lifecycle of applications. Threat Modelling is one such practice that outlines a structured approach to identify and classify threats within a software system. By successfully identifying, documenting and rating these threats, appropriate mitigations can be implemented in a more systematic manner.

Threat Modelling, when introduced early in the development lifecycle of new applications, greatly reduces mitigation costs through the early identification of vulnerabilities. Existing/legacy software systems can also benefit from Threat Modelling, although this research indicates that unique and complex challenges arise when dealing with historical/legacy systems, as opposed to new applications

This research presents different Threat Model methodologies and determines which approaches are most suitable for enterprise application development. Detailed comparisons of existing methodologies can benefit organisations considering introducing Threat Modelling into their development lifecycle.

Following this initial research, an in-depth case study Threat Model was developed for an existing legacy enterprise application. This case study investigated the practical implementation of the chosen methodologies and assessed the challenges of Threat Modelling a legacy system. In order to support this research, an Irish based software company provided access to an existing legacy system, and a Threat Model of that system was developed.

To this end the hypothesis of this research was deemed to be:

> STRIDE is a comprehensive Threat Modelling methodology for assessing software security vulnerabilities in legacy enterprise applications.

To evaluate the hypothesis two applicable threat model methodologies were applied to a mock application. The initial findings were presented in tabular format. Subsequent to this the STRIDE methodology was then applied to a legacy enterprise system. Verification of STRIDEs results was then presented. This research concludes with the presentation of the analysis of the findings with suggestions for further work.

CHAPTER 2.

BACKGROUND AND INITIAL RESEARCH

2.1 Background

One of the fundamental difficulties that the software industry has faced is security and the development of secure software. Traditionally with finite budgets, a development team's focus was primarily to create an application to meet specified functionality requirements. Security measures were implemented at a later stage in the Software Development Lifecycle (SDLC) with vastly varying degrees of competency, influenced by factors such as budget constraints and the skill-base of the security personnel. Based on industry experience, this author suggests that software was often developed and released with vulnerabilities easily exploited by adversaries with the necessary knowledge.

Over the last ten years, application development and the methodologies that govern development practices have become significantly more focused on security. In 2002 Bill Gates sent the now infamous "trustworthy computing" email to Microsoft employees, outlining the company's obligation to "lead the industry to a whole new level of Trustworthiness in computing" [1]. The security and privacy of data stored by software as well as security models, and clearer methods for developers to understand and build them into their applications, was at the core of this security improvement program. Microsoft was just one of many organisations at this time that was focusing on "building security in..." [2] to every aspect of the software development lifecycle.

Many development processes such as the Microsoft SDL (Security Development Lifecycle) have since been established to provide a structured approach to the development of more secure applications. Many of these development frameworks incorporate the practice of Threat Modelling. In its most basic form Threat Modelling is a structured approach that identifies potential security threats, assesses the risk and implements countermeasures.

2.2 Research

Each security development process discussed in this chapter incorporates Threat Modelling as part of its development framework.

This chapter outlines some of the leading development frameworks in secure application development including: Software Assurance Maturity Model (SAMM), the Building Security in Maturity Model (BSIMM) and the Comprehensive and Lightweight Application Security Process (CLASP) and Microsoft's Security Development Lifecycle (SDL). During initial research it was found that the SDL process provided a greater amount of supporting resources including documentation, tutorials and supporting software applications from the Microsoft website [3]. This wealth of information makes SDL an attractive option for organisations looking to adopt a new software security initiative.

The purpose of researching the different development frameworks was to gain a better understanding of software security, Threat Modelling, how it fits into software development, where it originated from and where Threat Modelling is going.

2.2.1 Open Software Assurance Maturity Model (SAMM)

OpenSAAM is an OWASP (Open Web Application Security Project) backed framework that is built upon a collection of security practices that are tied back into four critical business functions involved in software development [4], namely Governance, Construction, Verification and Deployment. Each business function has three security practices and each of these practices is divided into three maturity levels.

Threat Assessment is the first security practice during the Construction business function. It uses Threat Modelling to identify potential risks. OpenSAAM it not tied in to any one Threat Modelling approach and recommends the use of Microsoft's STRIDE or TRIKE as potential options.

2.2.2 Building Security in Maturity Model (BSIMM)

BSIMM security initiative was designed to help software development teams understand and plan security [5] into an applications development lifecycle by studying the practices of fifty-one leading software security initiatives. Companies such as Google, Adobe, Intel, Visa, Nokia, Sony and Microsoft [6a] participated in the Gary McGraw (an industry leading expert in software security) lead research. The resulting methodology combined the best practices (opinion of BSIMM team) into one initiative. Teodoro and Serrao [7] suggested that companies like Microsoft and Google give this security framework a special credibility.

Twelve practices grouped into four domains, Governance, Intelligence, SSDL Touchpoints, Deployment, are used to organise this software security framework's activities. The first practice within the Intelligence domain is Attack Models. Threat Modelling is used during this step to create an attack and data knowledge base [6b] relevant to the application.

BSIMM is not a new innovative approach to secure software development. This framework gathered data on which activities are actually performed by Industry leaders. It then promoted the best and most used of those activities. Traditionally, software firms were reluctant to divulge information on internal practices. Gathering this information is a positive achievement with relation to improving secure software development practices. BSIMM would benefit an organisation either looking to adopt a security initiative or improve/mature existing practices. For this research however BSIMM does not provide enough detailed information on Threat Modelling.

2.2.3 Comprehensive, Light-weight Application Security Process (CLASP)

CLASP, another OWASP backed security framework contains formalised best practices for building security into existing or new-start development life-cycles in a structured and repeatable way [8]. It has been in existence since 2005 but the project has seen no recent updates. It was originally developed by Secure Software Inc. and later donated to OWASP who released it as a complete security solution that organisations could implement. Along with

Microsoft's SDL, CLASP was recognised as one of the original [9] high profile processes for developing secure software.

Within the CLASP framework, seven best practices are the foundation of all security related activities. Performing an 'application assessment' is one of these seven best practices, which promotes a security analysis of systems requirements and design to be conducted using Threat Modelling. CLASP is not tied into any one method of modelling nor has it developed its own approach.

Due to the lack of evolution of this project, a more progressive framework would be preferable for a proposed security initiative.

2.2.4 Microsoft's Security Development Lifecycle (SDL)

The Security Development Lifecycle (SDL) is a product of Microsoft's "Trustworthy Computing" initiative. With Threat Modelling at its core it is a clear well documented software security development process with a number of software tools in support.

SDL aims to reduce software maintenance costs and increase reliability by building security into each stage of the development lifecycle. The process consists of security practices grouped by seven phases (Figure 1): training, requirements, design, implementation, verification, release, and response [10]. Microsoft has developed and released its own Threat Modelling methodology called STRIDE and different approaches to risk analyses including DREAD. Hussain, Erwin and Dunne [11] suggested STRIDE is regarded as the most widely used Threat Model methodology. The quantity of STRIDE based material identified during this initial research indicates this is an accurate statement.

> STRIDE is an acronym for:
> Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege.
>
> DREAD is an acronym for:
> Damage Potential, Reproducibility, Exploitability, Affected users, Discoverability.

At the end of the STRIDE activity a list of vulnerabilities is produced. These vulnerabilities are then ranked (low / medium / high) using a risk analyses technique such as DREAD. The end result is a prioritised list of vulnerabilities that both the development team and business stakeholders can discuss and use with relation to security improvements.

One of the keys to SDL is the introduction of Threat Modelling at the design stage which helps identify application vulnerabilities and even potential design flaws early. This information is then used to determine mitigations or potential design changes [2].

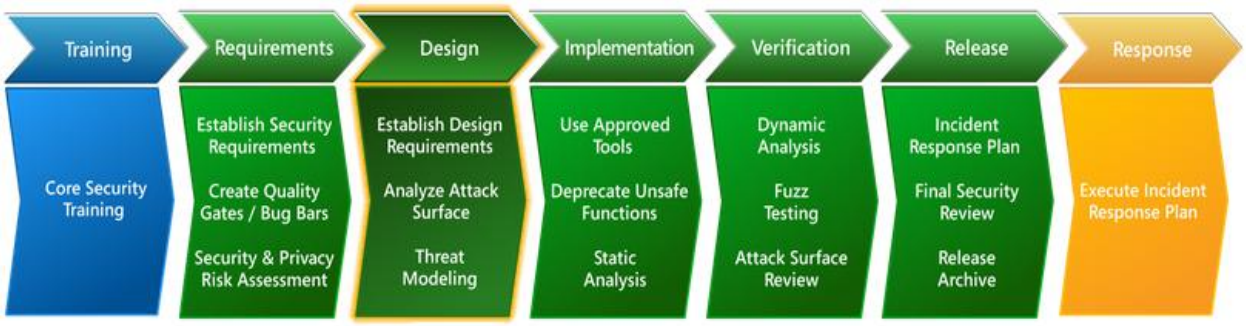


Figure 1 - Security Development Lifecycle Process [12]

It is evident during this research that Microsoft has put extensive resources behind its Security Development Lifecycle. The number of tools available, not just for Threat Modelling but every aspect of the SDL, online tutorials, documentation, support forums and blogs as well as the constant refining of the process demonstrates Microsoft's commitment to software security and the continuous improvement of the activities that are used to developed secure applications.

A recent IT threat evolution report by security firm Kaspersky Lab in Q3 2012 states that Microsoft products " ... no longer feature among the top 10 products with vulnerabilities" [13]. Traditionally Microsoft products would have had a heavy presence on any such list. Such findings provide evidence that SDL and the resources behind it are paying dividends.

Microsoft's SDL can be adapted to work with varying development environments. The process can be used with Waterfall and Agile development methodologies. It can also be applied to any platform and implemented by any size of organisation.

2.2.5 SDL for Agile

Further evidence of Microsoft's constant refining of its approach and commitment to secure software development was the release of SDL for Agile which is a customised version of the traditional SDL tailored to fit the agile software development cycle. Agile development is a popular alternative to more traditional development cycles such as the waterfall model. Development takes place in incremental, iterative cycles known as sprints [14]. A sprint is a short period of time (1, 2, 3 weeks, but can take longer) within which a set of features are developed and tested.

The changes to Microsoft's traditional SDL are focussed on cutting the standard framework requirements to a core subset to make it possible to complete within an agile development cycle [15]. The fundamental difference between the classic SDL and SDL for Agile is that each requirement necessary for classic SDL is not necessarily required for every Agile Sprint [16].

The SDL agile approach defines three categories of requirements. Each requirement is placed into one of these three categories based on the frequency of use of that requirement. The first category is the "Every-Sprint" level. These are for requirements that are necessary Every-Sprint, regardless how short that sprint may be. Threat Modelling is an Every-Sprint requirement as it is the cornerstone of the SDL. Threat Modelling can take a long time, so for SDL Agile only the current functionality that is being developed in that sprint is required to be Threat Modelled.

The second category is known as "On-Boarding" requirements. These are requirements that are completed only once at the beginning of the project. All projects require a baseline Threat Model for the application. If it is a new project the development team only need to model features as they are designed. If the development team of an existing legacy system wishes to move to the SDL Agile approach, then a baseline Threat Model must be built of that system.

All remaining SDL requirements are placed within "Bucket" requirements. These are tasks that are performed on a regular basis over the lifetime of a project. This category is subdivided into three bucket requirements: Security Verification, Design Review and Planning.

Microsoft has adapted SDL successfully to fit the Agile Development methodology. One aspect of interest in Agile SDL is the requirement of a baseline Threat Model if applying to an existing system. Similarly, converting an existing project to the classic SDL approach also requires a Threat Model of the existing system. Little documentation is available identifying the specific challenges of Threat Modelling such systems. Legacy projects may no longer have original developers within the team. Aspects of the code base and architecture may not be sufficiently documented. In Chapter 8, the historical nature of a legacy project upon a Threat Model is also considered.

In 2010, an Errata security survey identified SDL as the most frequently employed technology by teams implementing software security frameworks. However, the report does not discriminate between new and legacy projects. Therefore, a more in-depth review is required for accurate information on the success of methodologies most suitable for security modelling.

2.2.6 Security survey reported SDL / SDL for Agile most adopted processes

In early 2010 Errata Security conducted a survey which gathered information from forty six companies on current security practices and software development methodologies such as SDL, SAMM, and BSIMM [17]. Its aim was to establish which organisations were implementing these methods, as well as to discover the reasons why companies are abstaining from these methods.

The survey's findings reported that Microsoft's SDL was the most adopted security development practice by organisations, with SDL for Agile the second most popular. 35% of responses stating they use SDL for Agile [18]. Another interesting finding is that 81% of firms are aware of formal secure software development efforts such as SDL, CSIMM, SAMM and CLASP [2]. The survey also suggests that companies not deploying security methodologies are identifying the lack of resources as the reason.

As this survey is based on a very small data set, the findings can only be described as indicative. However, they do provide a promising indication of the popularity of SDL/SDL for Agile. While it would have proved interesting to do an up-to-date version of this survey it is also important to

remember that many companies are reluctant to discuss security techniques because of the risk of exposure and/or attack.

2.3 Conclusion

The range of frameworks that now include Threat Modelling suggests that the inclusion of Threat Modelling is becoming an industry best practice for identifying security vulnerabilities in software.

This research has also established Microsoft's SDL as an industry leader in secure software development with Threat Modelling at its core. A considerable number of resources are available for organisations that adopt SDL. Large enterprises also tend to favour well established software houses such as Microsoft. This may explain the results of the Errata Security Survey which identified SDL as the leading security development framework.

It is worth noting that limited information was identified outlining the challenges of maintaining or further developing legacy enterprise system. The case study presented in Chapter 6 of this research focuses on the Threat Modelling of a large scale legacy system. More detailed research was performed focusing on individual Threat Model methodologies available (presented in Chapter 4) for adoption by enterprise development companies.

This Chapter discussed leading development frameworks and the position of Threat Modelling within those frameworks. Initial research indicates the importance of establishing the role of Threat Modelling in security software development. Threat Modelling and the different methodologies available are discussed in further detail in Chapter 3.

CHAPTER 3.

THREAT MODELLING

This Chapter discusses basic Threat Modelling concepts, identifies different methodologies including STRIDE and TRIKE and the supporting tools available for these Threat Models. An overview of Risk Assessment is also provided.

Designing secure software applications can be difficult, but with any challenge, a good strategy is to break the problem down into small parts that are more easily managed. This is one of the fundamental goals all Threat Models have in common. Threat Modelling involves identifying assets, identifying potential threats to those assets, categorising the threats and identifying mitigation strategies in a structured process.

3.1 Threat Modelling Basic Concepts

When beginning research into Threat Modelling it is important to have an understanding of basic terminology.

- An **Asset** is something of value that an adversary requires. The data in a database is an example of an asset.
- A **Threat** is an event which may or may not be malicious in origin that can damage or compromise an asset by performing an attack.
- A **Vulnerability** is a flaw in some part of systems security that makes a threat possible.
- An **Attack** is an attempt by an adversary to exploit a vulnerability.
- A **Risk** is the likelihood of being the target of an attack, the attack being successful and the impact if successful.
- A **Countermeasure** is an action or tool that addresses a threat and mitigates the risk.

3.2 Introduction to Threat Modelling

Threat Modelling can be defined as the “Systematic review of features and product architecture from a security point of view” [19]. The process provides a structured approach to identifying and classifying threats based on the software’s components, data flows and different trust boundaries. Threat Modelling is an important part of the software development lifecycle as it identifies threats that would potentially be missed by traditional security brainstorming sessions. Unlike software testing techniques such as penetration testing or Fuzzing, Threat Modelling can be performed during a systems design phase making it code independent. Introduced early in the development lifecycle Threat Modelling can help ensure an applications design is secure and reduce costs by minimising required security fixes late in the project. Existing systems can also benefit from the process. Unknown or unmitigated security issues can be identified in legacy systems and risk rating applied to these vulnerabilities identified. The process can also be tailored to fit development practices such as Agile.

Initial research indicated that few detailed comparisons have been published comparing different processes such as STRIDE and TRIKE. This research presents some in-depth comparisons of selected methodologies most suitable for enterprise application development (Chapter Four).

3.3 Microsoft SDL – STRIDE

The goal that all Threat Modelling methodologies share is the development of a process of iterative steps that a development team can easily follow when evaluating a software system. Figure 2 depicts the main steps in preparation and performing a STRIDE threat model. STRIDE supports the identification and categorisation of threats in a software application [20]. Methodologies such as STRIDE successfully reduce the number of threats to software applications. A repeatable structured approach to software security allows non-security personnel to also participate in the process.

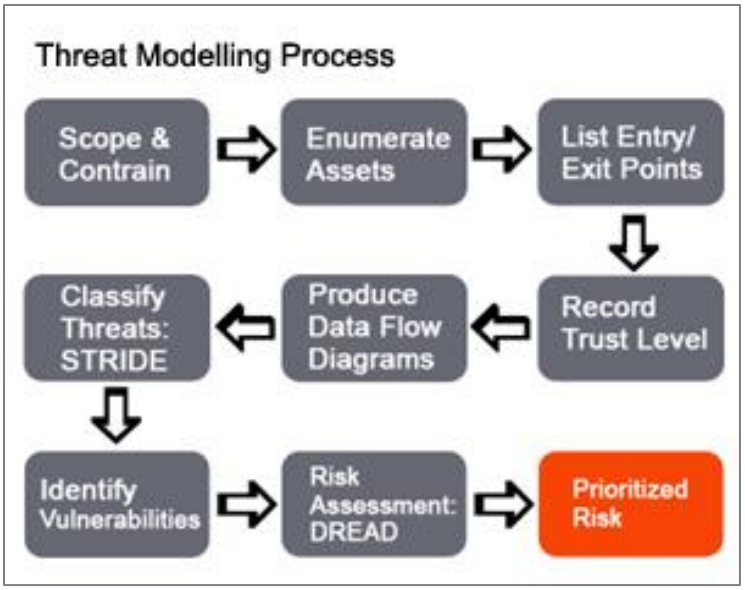


Figure 2 - SDL Threat Modelling Steps

The precise processes of Threat Modelling methodologies can differ, but all of them involve the collection of information about the application, its environment, how it is used and deployed, how vulnerabilities are identified, and risks assessed. Some of the steps that the Microsoft SDL approach (Figure 2) following include:

3.3.1 Scope and Constraint

Threat Modelling can be a time consuming process especially if modelling an existing enterprise scale application. Clear objectives help focus the Threat Model activity [21]. The scope of the model should be defined at the beginning. In the case of larger systems, more value can be quickly reaped by constraining the process to specific areas.

At the commencement stage of the Threat Model process, when scope and constraint are defined, additional information such as Use Scenarios, Dependencies and Security Notes may also be recorded to help build up a better understanding of system.

3.3.2 Use Scenarios

A list of usage scenarios or stories that demonstrate the intended use of the application is created. This step can aid in the identification of potential areas for system abuse.

3.3.3 External / Internal Dependencies

Software applications often depend upon other systems or components. These other components can have a direct impact on the security of a system. It is important to list these dependencies and identify what impact they may have on a system. Examples include: Antivirus, Firewalls, Third Party libraries and Authentication Systems.

3.3.4 Security Notes

Details such as security-relevant information already known about a system, assumptions made about the application, or design trade-offs, are recorded. These security notes can assist decision making.

3.3.5 Assets

The assets that may be of interest to a potential adversary are listed. They may be a physical or logical asset. Examples of potential assets include Login credentials, Credit Card details, Encryption keys or User details.

3.3.6 Entry / Exit Points

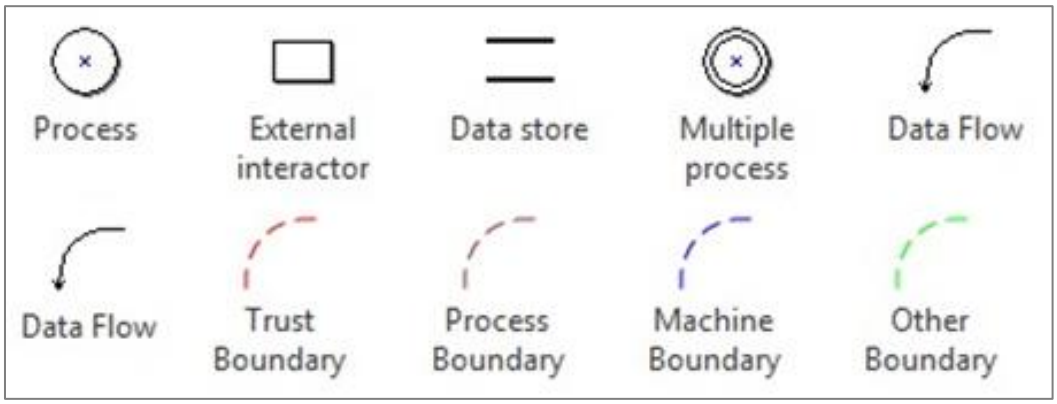
Entry/Exit points are where the data enters or exits the system. These are sometimes referred to as attack points [22] as this is where a potential attack can occur. Each of these input/output points correspond to a location in a system where security measures should be implemented.

3.3.7 Trust Levels

A trust level is used to define the privileges an external entity has to gain access to a system. They can be categorised according to privileges assigned or credentials supplied, and cross referenced with entry/exit points and protected resources. Examples include anonymous user/system, authenticated user/system and the administrator. Trust levels are applied at each entry/exit point.

3.3.8 Architecture Overview – Data Flow Diagrams

Threat Modelling focuses on the data, how it flows and how it moves between components. Modelling diagrams provide a visual representation of how the subsystems operate and work together. Data Flow Diagrams (DFD's) are a popular choice for many Threat Modelling processes. They provide a compact and consistent way to model data flows in an application with the use of six shapes that represent: Process, Multiple Process, External Entity, Data Store, Data Flow, and Privilege Boundaries (Figure 3).



Note: Screen shot from threat modelling tool

Figure 3 - DFD Shapes [23]

Privilege boundaries and further defined. Trust boundaries are identified by a red dashed line, Process boundaries identified by a brown dashed line, Machine Boundaries identified by a blue dashed line and Other Boundaries identified by a green dashed line. The use of colour allows teams to easy recognise which type of boundary data flow is crossing when assessing system DFDs.

3.3.9 STRIDE Threat Classification

Identifying potential threats against the system modelled is one of the main goals of the Threat Modelling process. By understanding the threats it is possible to determine an application's vulnerabilities. Microsoft developed the STRIDE model for identifying and classifying threats into categories. STRIDE (Table 1) is an acronym for: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

Threat Type	Description
Spoofing	Allows an adversary to pose as another user, component, or system that has an identity in the system being modelled
Tampering	The modification of data within the system to achieve a malicious goal
Repudiation	The ability of an adversary to deny performing some malicious activity because the system does not have sufficient evidence to prove otherwise
Information disclosure	The exposure of protected data to a user that is not otherwise allowed access to that data
Denial of Service	Occurs when an adversary can prevent legitimate users from using the normal functionality of the system
Elevation of Privilege	Occurs when an adversary uses illegitimate means to assume a trust level with the different privileges than he currently has.

Table 1 - STRIDE Model

Microsoft originally applied STRIDE to each entry / exit point. This approach was refined to apply STRIDE against each element in the applications Data Flow Diagrams (Table 2).

DFD Element	S	T	R	I	D	E
External Entity	X		X			
Data Flow		X		X	X	
Data Store		X	*	X	X	
Process	X	X	X	X	X	X

***applies if the data store contains logging or audit data**

Table 2 - STRIDE-per-Element [24]

For example a data flow between two components, a process and a data store, under the classification of STRIDE (Table 2) is potentially vulnerable to Tampering, Information Disclosure and Denial of Service. The Threat Modelling process works through each threat per-element and assesses if proper mitigation techniques are in place for that threat type. By working

through each element of the system as identified in the DFDs, a threat profile of the application is created. Each threat is either mitigated or accepted. For none security experts STRIDE provides classifications of mitigation techniques (Appendix 1 – STRIDE mitigations) for each potential threat type identified (Table 3).

Threat Type	Mitigation Technique
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation services
Information disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Table 3 - Mitigation Techniques [25]

One obvious benefit of the STRIDE per element approach is code independence. This is advantageous as it helps identify security issues at design stage. It also allows members of the team other than developers to participate in the Threat Model.

The process can benefit small/medium sized enterprises (SME's) which may not have a security expert within their team. STRIDE allows such teams to identify vulnerabilities and the mitigation techniques that can defend against vulnerabilities. Larger organisations can also benefit from the process for both new projects as well as existing projects where unidentified vulnerabilities may exist within legacy code. Implementing a methodology that identifies and classifies threats is a structured repeatable process can beneficial all project types. The end result is a list of vulnerabilities that development teams and product stakeholders can then assess. At this point, an accurate risk assessment of vulnerabilities may be made. This topic is discussed in section 3.8, Risk Assessment.

STRIDE has been identified as an industry leading Threat Modelling methodology in the software industry. The success of this methodology is because of a well structured approach to Threat Modelling, and excellent support and resources for users. Other Threat Modelling methodologies are also available to organisations such as TRIKE, P.A.S.T.A and practices based on AS/NZS 31000:2009.

3.4 TRIKE overview

TRIKE is an open-source framework for security auditing from a "risk management perspective through the generation of Threat Models in a reliable, repeatable manner" [26]. The project began in 2005 as an attempt to improve the efficiency and effectiveness of existing Threat Modelling methodologies and is being actively used and developed since [27a]. The creators of TRIKE have also developed tools to accompany this methodology such as the TRIKE spreadsheet. This tools focus is to automate threat generation. There is no brainstorming involved. Security teams do not need to think up potential threats as the threats are Pre-defined. An inexperienced security developer can use TRIKE and reliably find security vulnerabilities.

The TRIKE team have adopted HAZOP (Hazardous Operations) analysis, "A systematic method, for identifying which variations in a process need to be mitigated" [27b]. This process replaces threat and attack trees.

TRIKE uses a risk based approach with distinct implementation, threat and risk models. It approaches Threat Modelling from a defensive position as opposed to that of an attacker's. TRIKE has also proposed threat chaining, an alternative to threat trees, in an attempt to reduce the repetitive nature of threat trees.

It was not the intention of this research to Threat Model a complete system using different approaches, but a practical comparison was deemed beneficial. During research no adequate material comparing TRIKE with STRIDE was identified. As stated, TRIKE approaches Threat Modelling from a defensive approach when compared to STRIDE. It was for this reason TRIKE was researched further and a comparison with STRIDE performed.

3.5 P.A.S.T.A (Process for Attack Simulation and Threat Analysis) overview

This is a seven step process that is applicable to most development methodologies and is platform agnostic. Its main aim is focussed at addressing the most viable threats to a given application target [28].

Seven steps of P.A.S.T.A

- Define Business Objectives
- Define Technical Scope
- Application Decomposition
- Threat Analysis
- Vulnerability Detection
- Attack Enumeration
- Risk/Impact Analysis

The process is a combination of various Threat Modelling approaches, defines the business objectives, security and compliance requirements and business impact analysis. Similar to the Microsoft process, the application is cut down into components with use cases and DFDs used to illustrate the application. Threat trees and abuse cases are also used during threat and vulnerability analysis. Risk and business impact are calculated and the necessary countermeasures identified.

During this research P.A.S.T.A was identified as a possible Threat Model methodology for further research. When compared to TRIKE and STRIDE it lacked the necessary resources such as documentation and example Threat Models using P.A.S.T.A. For this reason it was not considered for this research case study.

3.6 AS/NZS 31000:2009 Risk Management overview

The AS/NZS 31000:2009 issued in November 2009 was built on the Australian/New Zealand Standard AS/NZS 4360:2004, the world's first formal standard for documenting and managing risks and provides generic guidelines on risk management [29]. The standard approach is

simple, flexible and iterative. It does not lock organizations into a particular risk management methodology, provided they fulfil the five steps of the AS/NZS 4360 process.

- Establish Context: Establish the risk domain, by defining what assets/systems are important.
- Identify the Risks: Within the risk domain, what specific risks are apparent?
- Analyze the Risks: Looks at the risks and determine if there are any supporting controls in place.
- Evaluate the Risks: Determine the residual risk.
- Treat the Risks: Describes the method to treat the risks so that risks selected by the business will be mitigated.

AS/NZS 31000:2009 Risk Management may be used to rank risks for security reviews. However, because of the lack of structured methods regarding threat enumeration, it is a less desirable model and thus dismissed as unsuitable for this case study.

3.7 Other Threat Model processes

There are many other Threat Modelling methodologies developed to assist security personnel. One example of this is the Practical Threat Analysis (PTA), a calculative Threat Modelling methodology and suite of software tools [30]. The methodology chosen by an organisation depends on their individual security requirements and objectives and in many cases can involve a combination of different practices which best suit that organisation's needs.

3.8 Risk Assessment

For each threat identified as a result of the threat identification process such as STRIDE, it is necessary to check if the required mitigation techniques are in place. Threats without mitigation are vulnerabilities. All unmitigated threats identified manifest a vulnerability list. A Risk Assessment, which is the final step of the Threat Modelling process, is implemented to prioritise each vulnerability. This step focuses on aspects including: Severity, Ease of

exploitation by a potential attacker as well as the Impact/Cost of a successful exploitation of a vulnerability.

Microsoft has currently[1] developed two such approaches; DREAD and the Security Bulletin Severity Rating System.

3.8.1 DREAD

Microsoft developed the DREAD (Table 3) methodology to rate each risk identified during the STRIDE activity. Each risk is assigned a DREAD score by the security/development team performing the Threat Model. Different variations of the scoring system exist.

DREAD	Description
Damage potential	Ranks the extent of the damage that occurs if a vulnerability is exploited
Reproducibility	Ranks how often an attempt at exploiting a vulnerability works
Exploitability	Assigns a number to the effort required to exploit the vulnerability. In addition, exploitability considers preconditions such as whether the user must be authenticated
Affected Users	A numeric value characterizing the ration of installed instances of the system that would be affected if an exploit became widely available
Denial of Service	Measures the likelihood that a vulnerability will be found by external security researchers and hackers if it went un-patched

Table 4 - DREAD Model

When assessing risk each DREAD component is given a score. The components scores are then calculated to give a total 'DREAD Score'. The risk is then determined by the score range the 'DREAD Score' falls into. The final outcome is a list of vulnerabilities ranked by risk.

The process of applying DREAD requires the most opinion and expertise. It is recommended to have at least one team member who is familiar with security to assist in compiling the DREAD scores. The practical implementation of DREAD is discussed in detail in Section 4.1.

[1] June 2013

3.8.2 Security Bulletin Severity Rating System (S.B.S.R.S)

Microsoft also has the 'Security Bulletin Severity Rating System' as an alternative to DREAD to rate vulnerabilities in Microsoft products. The rating system places vulnerabilities into one of four categories (Table 5). Microsoft uses this system to rate the necessity of security patches for its products.

Rating	Definition
Critical	A vulnerability whose exploitation could allow code execution without user interaction.
Important	A vulnerability whose exploitation could result in compromise of the confidentiality, integrity, or availability of user data, or of the integrity or availability of processing resources.
Moderate	Impact of the vulnerability is mitigated to a significant degree by factors such as authentication requirements or applicability only to non-default configurations.
Low	Impact of the vulnerability is comprehensively mitigated by the characteristics of the affected component. Microsoft recommends that customers evaluate whether to apply the security update to the affected systems.

Table 5 - Security Bulletin Severity Rating System [31]

This rating system can also be adapted to rate vulnerabilities identified by STRIDE. The end goal is a prioritised list of vulnerabilities that will assist decision making. Although Microsoft no longer uses DREAD, it is the opinion of this author that it is a more suitable risk assessment process for non-security experts or teams new to Threat Modelling compared to the Security Bulletin Severity Rating System. DREAD performs a calculation based on assessing different aspects of the vulnerability such as Damage Potential or Reproducibility. The Security Bulletin Severity Rating System places vulnerabilities into one of four categories. Both are based on the opinion of the individual/team performing the risk assessment. DREAD has refined its scoring system reducing the likelihood of varying results and offers a structured approach for teams to adopt. The process is also well documented and relatively easy to implement. For these reasons DREAD was chosen over S.B.S.R.S.

3.8.3 Other Risk Assessments Processes

There are other options available for risk assessment such as US-CERT Vulnerability Metric which uses a quantitative metric and scores the severity of a vulnerability by assigning to it a value between 0 and 180. The Common Vulnerability Scoring System (CVSS) [32] aims to provide an open framework for measuring the impact of IT vulnerabilities while the SANS Critical Vulnerability Analysis Scale ranks vulnerabilities using several key factors and varying degree of weight. The risk assessment process to adopt comes down to the individual choice of the Threat Model team.

3.9 Threat Modelling supporting tools

There are a number of tools available to assist the Threat Modelling process. This section includes a summary of the tools encountered during research. Some of these tools are developed for a specific methodology. Microsoft have released several tools to assist their STRIDE methodology include SDL Threat Modelling (TM) Tool, its predecessor Threat Analysis and Modelling (TAM) v3.0 and SDL Elevation of Privilege (EoP). The first two tools incorporating Viso [33] which simplifies the drawing of Data Flow Diagrams and provides the ability to check diagrams against business rules and logic to ensure accuracy and consistency. These tools apply STRIDE to each element of the DFD and provide guidance throughout the Threat Modelling process.

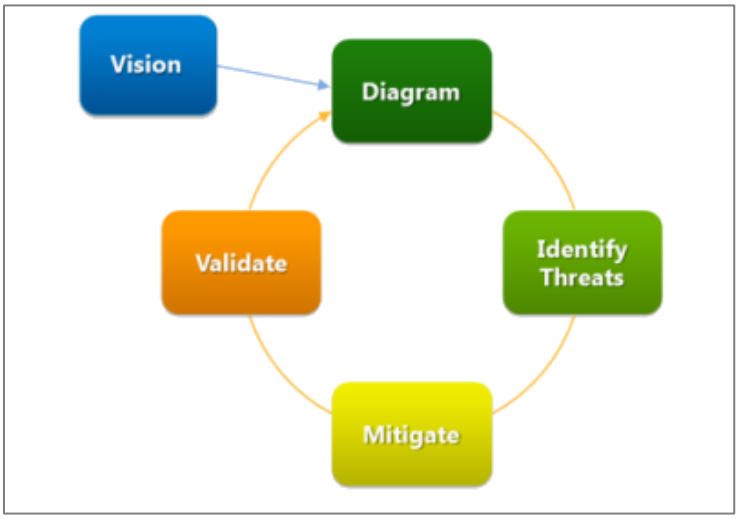


Figure 4 - SDL Threat Modelling Tool Process [34a]

The SDL TM Tool is the most recent[2] release which refines the process (Figure 4). The focus of the analysis is to diagram the system, identify threats using STRIDE, mitigate these threats and validate by repeating the process. SDL EoP is a card game to assist I.T personnel get started with Threat Modelling. EoP attempts to clarify the details of Threat Modelling and examine possible threats to software and computer systems. It uses a simple points system that allows you to challenge other developers and become your opponent's biggest threat [34b]. It is a good starting-point to Threat Model any system. And can be used by a security teams to commence initial analyses.

The TRIKE team have two tools to support their methodology. The first is a desktop application that auto generates threats and attacks tree stubs. This tool is mildly buggy and outdated. The TRIKE spreadsheet is the most current tool. The majority of teams that use TRIKE do so with the assistance of this tool [35]. It is a very complex spreadsheet that has the TRIKE rules built in. It auto-generates the threats, prioritises these threats, records security objectives and has data collection for the privilege analysis as well as support for HAZOP analysis.

The Threat Modelling process can be greatly assisted by supporting tools. As part of this research the SDL Threat Modelling tool and TRIKE's spreadsheet were used when assessing TRIKE v STRIDE.

3.10 Conclusion

Threat Modelling provides a structured, repeatable approach to identify, prioritise and mitigate vulnerabilities in software systems. Information presented in this Chapter include: Threat Modelling concepts, Threat Modelling methodologies and software tools developed to support the practice.

The Threat Model methodologies reviewed in this Chapter were chosen for two reasons. The first is that they were methodologies identified in the initial research into security development processes. Secondly, they were approaches identified as possible candidates for

[2] June 2013

implementation in the case study presented by this research. Two of these methodologies were chosen for comparison by implementation (Chapter 4), STRIDE and TRIKE. This research has identified SDL's STRIDE as a well-structured approach to Threat Modelling with supporting tools, tutorials and documentation easily available. These are characteristics that may make this approach more appealing to enterprises that are considering the introduction of Threat Modelling into their development lifecycles.

When adopting a new methodology, organisations consider the number resources the implementation of the methodology require, the improvements to current practices it brings, how difficult it is to use and the long term benefits to the project. STRIDE can improve software security, can be introduced with minimal overheads, can begin identifying vulnerabilities quickly, is relatively easy to implement and provides the long term benefit of producing more secure software. For these reasons STRIDE was considered for comparison with TRIKE.

TRIKE approaches Threat Modelling from a defensive, risk management perspective. This may appeal to enterprises as developers traditionally are focused on functionality. TRIKE focuses on securing an application's features, compared to methodologies such as STRIDE which require Threat Modelling teams to assess a system from the view point of an adversary. TRIKE also has documentation and supporting tools readily available. Enterprises may consider TRIKE as it can help identify threats quickly, is a free open source methodology and tool, improves software security and is progressive with the TRIKE team continuously refining its process and tools. For these reasons TRIKE was considered for comparison with STRIDE.

CHAPTER 4.

COMPARISON OF THREAT MODEL APPROACHES

In preparation for performing a Threat Model on an enterprise sized application, different approaches were evaluated during this research. A mock web application (Figure 4) was threat modelled using two methodologies, STRIDE and TRIKE. The STRIDE approach was performed both manually and with the aid of the SDL Threat Modelling tool (v3.1.8). TRIKE v1.5 with supporting spreadsheet was also implemented.

This mock application represents a standard website with user and administration login functionality to restricted content. The focus of this exercise was to evaluate and compare the different Threat Modelling processes by working through the different approaches.

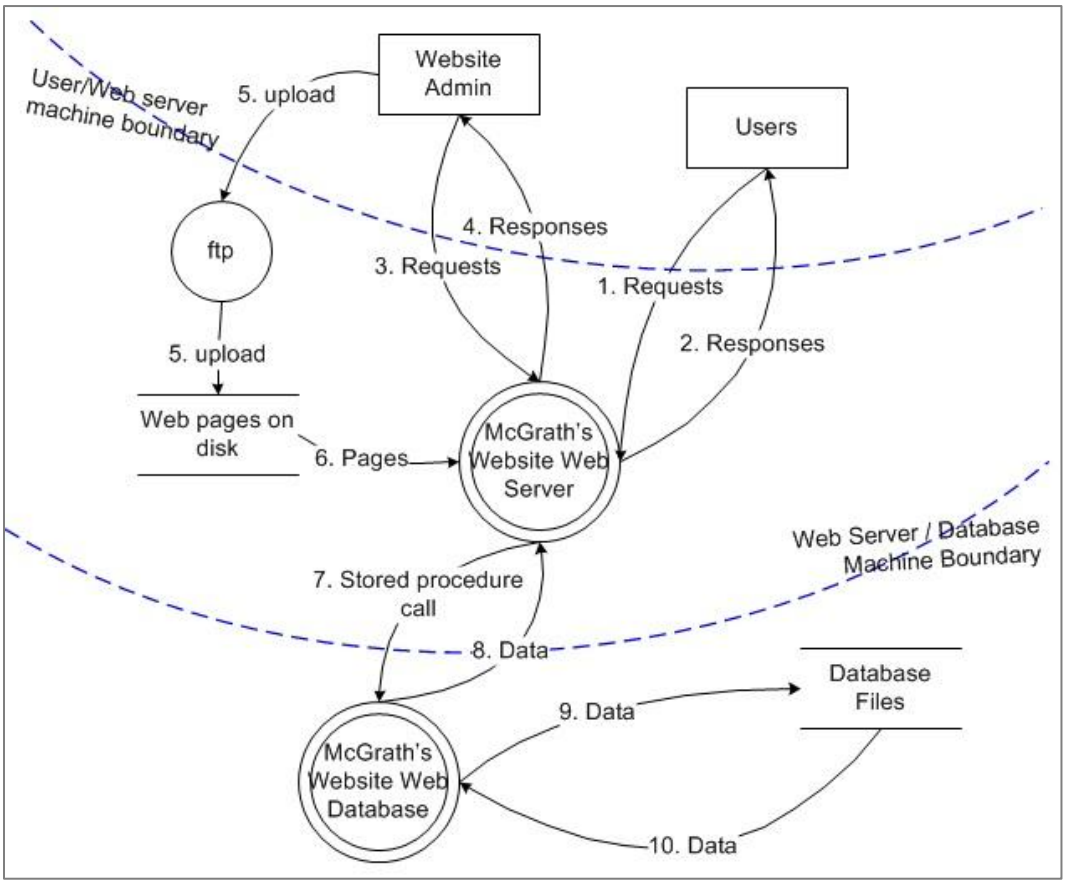


Figure 5 - Context DFD of McGrath's Mock Web Application

4.1 STRIDE / DREAD

Firstly the mock web application was modelled using the traditional STRIDE / DREAD approach (Appendix 2 – STRIDE Threat Model) outlined in Chapter 2. This was performed manually to demonstrate that STRIDE is not dependent on its supporting tools. The initial steps defined the scope of the model, identified use scenarios, listed dependencies, security notes, identified assets, identified entry / exit points and listed user trust levels. The architectural overview was provided with the use of DFDs and STRIDE was applied to each element in these DFD's.

An analysis team at this stage would work through each threat with the assistance of the STRIDE mitigation worksheet. This step determines which elements have been protected with the correct mitigations and therefore not vulnerable. The final output was a list of vulnerabilities.

DREAD was used to perform a risk assessment. The vulnerabilities listed in the STRIDE activity where assigned a DREAD score based on the severity/impact in each of the dimensions of DREAD. The final output was a list of vulnerabilities with corresponding risk ratings which can be used to prioritise which vulnerabilities are mitigated first.

Performing STRIDE manually on a small mock application was time consuming. This would indicate that performing STRIDE manually on an enterprise scaled application would be a long process requiring a lot of time and resources allocated. This would make the methodology less appealing for adoption by an organisation. The manual method did however provide the opportunity to document a significant amount of data about the application. This would be of benefit for legacy systems which may not have significant documentation. The gathering and documenting of this data would possibly benefit future product enhancements.

4.2 SDL Threat Modelling Tool (v3.1.8)

The SDL Threat modelling tool is a Microsoft released application that provides a STRIDE based approach to create and analyse Threat Models and was applied to the mock application

(Appendix 3 – TM Tool). This tool automated the STRIDE process but does not implement DREAD risk assessment.

This tool allows for impact assessment, proposed mitigations and the generation of reports to ensure vulnerabilities are mitigated or eliminated [36]. This approach divides the Threat Model process into four steps:

i. Draw Diagram
ii. Analyse Model
iii. Describe Environment
iv. Generate Reports

4.2.1 Draw Diagram

DFD's are central to the SDL Threat Model (TM) tools approach. The Microsoft drawing application 'Visio' is embedded which allows for easy drag and drop functionality when creating DFDs. The TM tool takes the process one step further by providing automatic validation of each diagram. This validation can identify potential design flaws early in an applications development. If the design is large the tool also allows for multiple child diagrams that are related hierarchically. This assists with legibility which proved particularly useful in the modelling of the enterprise size case study.

4.2.2 Analyse Model

The analysis of the systems diagrams is based on the STRIDE-per-Element Model which provides a prescriptive approach to threat modelling [36]. When modelling of the system is complete, the Threat Modelling tool automatically analyses the design and produces a list of threats against each element in the systems. For example each external entity will have spoofing and repudiation threats listed (Figure 6) as these are the STRIDE classification of threats to which external entities are vulnerable.

ID	Element Name	Element Type	Element Diagram References	Threat Type	Bug ID	Completion
86	7. Stored procedure call (McGra...	DataFlow	McGrath Web app Context - Level dataflow diagram	InformationDisclos...		
87	7. Stored procedure call (McGra...	DataFlow	McGrath Web app Context - Level dataflow diagram	DenialOfService		
88	8. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	Tampering		
89	8. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	InformationDisclos...		
90	8. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	DenialOfService		
74	9. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	Tampering		
75	9. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	InformationDisclos...		
76	9. Data (McGrath's Website We...	DataFlow	McGrath Web app Context - Level dataflow diagram	DenialOfService		
70	Database Files	DataStore	McGrath Web app Context - Level dataflow diagram	Tampering		
71	Database Files	DataStore	McGrath Web app Context - Level dataflow diagram	Repudiation		
72	Database Files	DataStore	McGrath Web app Context - Level dataflow diagram	InformationDisclos...		
73	Database Files	DataStore	McGrath Web app Context - Level dataflow diagram	DenialOfService		
95	Web pages on disk	DataStore	McGrath Web app Context - Level dataflow diagram	(NotGenerated)		
33	Users	Interactor	McGrath Web app Context - Level dataflow diagram	Spoofing		
34	Users	Interactor	McGrath Web app Context - Level dataflow diagram	Repudiation		
35	Website Admin	Interactor	McGrath Web app Context - Level dataflow diagram	Spoofing		
36	Website Admin	Interactor	McGrath Web app Context - Level dataflow diagram	Repudiation		
64	McGrath's Website Web Database	MultiProcess	McGrath Web app Context - Level dataflow diagram	Spoofing		

Figure 6 - Threat Listings

Each threat instance is provided with an individual analysis screen where details such as threat impact and mitigation details can be recorded. The analysis of each threat remains a manual process, however this tool provides great assistance in tracking each step of that process as well as providing bug tracking functionality. This tool also provides a completion status bar to indicate the progress of the four stages: threat, mitigation, bug filed and complete.

4.2.3 Describe Environment

The security of any application must also consider the environment in which it is deployed. Information gathered does not directly impact the analysis of an application, but must be assessed and documented in order to complete the Threat Model process. Details such as third party dependencies and external security notes are recorded. This often proves difficult in an enterprise-sized application.

4.2.4 Generate Reports

Once all information is gathered and correctly entered, five report types can be generated and displayed: Bug, Analysis, Threat Model, Diagram Only and Recommended Fuzzing.

The Bug report lists all bugs entered against threats. The Analysis report shows the status of various elements, threats and certifications identified in the model. Threat Model Report provides a comprehensive report containing all the information captured during the Threat Model. The Recommended Fuzzing provides a priorities list of recommended targets for Fuzz testing, a software testing technique that provides invalid, random data to the inputs of a program. Reports on an enterprise system can be unyielding and may be ignored. Prioritising actions based on these reports is vital.

4.3 SDL Threat Modelling Tool vs. Traditional STRIDE / DREAD

The SDL Threat Modelling Tool automates many of the steps in the Threat Modelling process but fundamentally produces the same STRIDE-per-element analysis of the system based on the systems DFDs. The TM Tools approach has several advantages over a manual approach:

- Requires less documentation early in the Threat Modelling process. Analysis is performed earlier, potentially identifying vulnerabilities sooner.
- Diagram validation is automatically provided as each DFD is constructed.
- The analysis model auto-generates the STRIDE-per-element vulnerability listing.
- The analysis model also provides a centralised location to assess threat impact, record mitigations, bug tracking and provides a completion status for each vulnerability.
- Generates Reports.

Apart from automating the process the main differences include; systems documentation recorded and the risk assessment process. Traditionally information such as use scenarios, external dependencies, security notes, trust levels, assets and entry / exit points were documented before DFD's were created and STRIDE implemented. This information does not affect the analysis of the systems design. It can also be a time consuming process, especially on

enterprise scale applications. Information such as Entry / Exit points are traditionally documented, however they have no impact on the systems analysis, as STRIDE is implemented against each DFD element and not against documented system characteristics. The Threat Modelling tool has streamlined the required documentation from a security perspective. It records information that describes the environment which the application is deployed: Dependencies, Assumptions and External Security notes.

The TM tool does not provide risk rating or DREAD rating features. The reason for its exclusion was the traditional DREAD rating scoring system was too subjective. A security oriented member of the analysis team could rate a particular threat as high, awarding 9 or 10 on a scale of 1-10. Developers not as security conscious or business personnel could rate the same threat with low rating, awarding 2-3. For this reason Microsoft has moved away from DREAD.

DREAD scoring has been refined to reduce the polarity in risk ranking. Each vulnerability identified during the STRIDE activity is assigned a grade 1-3 (1 = low, 2 = Medium, 3 = High). The individual scores are totalled to provide a DREAD score.

Determine the risk:

DREAD Score: 5 - 8 = Risk Rating: Low
DREAD Score: 9 - 12 = Risk Rating: Medium
DREAD Score: 12 - 15 = Risk Rating: High

Ranking risks provides Threat Modelling teams the ability to prioritise which vulnerabilities to mitigate first. This is especially important when teams face financial constraints. The traditional DREAD approach produced varying results. A refinement of this approach reduces variations and offers teams the ability to produce a more accurate risk rating. This is crucial in enterprise development to validate the results obtained so that resources may be most appropriately applied when mitigating risks.

4.4 TRIKE 1.5

TRIKE has developed a very complex spreadsheet that relies on formatting to convey a considerable amount of information. Each tab in the spreadsheet is used to describe/identify different aspects of the system being modelled. The use of colour is prominent throughout; with each colour carrying a different meaning. For example, a red cell highlights a risky setting/decision. If the “Intended Actions” tab is red, this means a user should not be able to complete the action.

All Threat Model processes involve a number of steps that analyse the system, identify threats, identify mitigations and document decisions. TRIKE starts performing these steps at requirements level. TRIKE differs from STRIDE when performing analysis on the systems architecture. It prunes analysis based on the Threat Models security objectives, defined at the beginning of the TRIKE Threat Model process. This refining of analysis occurs after threats are identified but before they are investigated and again when threats are investigated but before any mitigation’s are identified.

The spreadsheet (Appendix 4 – TRIKE Spreadsheet) provides a number of tabs, each representing a required step. The most important of these tabs include:

- Overview – General meta data about the system
- Actors – System components and privileges
- Data Model – Data exchanged with or manipulated by the system
- Intended Actions – Systems business logic / functional requirements
- Connections – physical or logical between actors
- Threats – Summary of actions that violate the systems business rules
- Security Objectives – Ultimate security goals
- Use Cases - describes how the system implements intended actions that are likely to affect the threats in the security objectives, as well as HAZOP analysis.

The system is described by assets, while actors are the intended users which are also referred to as favoured users. An attacker can also be a 'Favoured user'. The focus is not on motivation or skill levels of an adversary, but on what privileges a user starts with and ends with after using the system. TRIKE focuses on the taxonomy of CRUD (create, read, write, update, delete) for every action that is possible on each asset (Figure 7). Additional actions, 'Execute' and 'Configure' are also available.

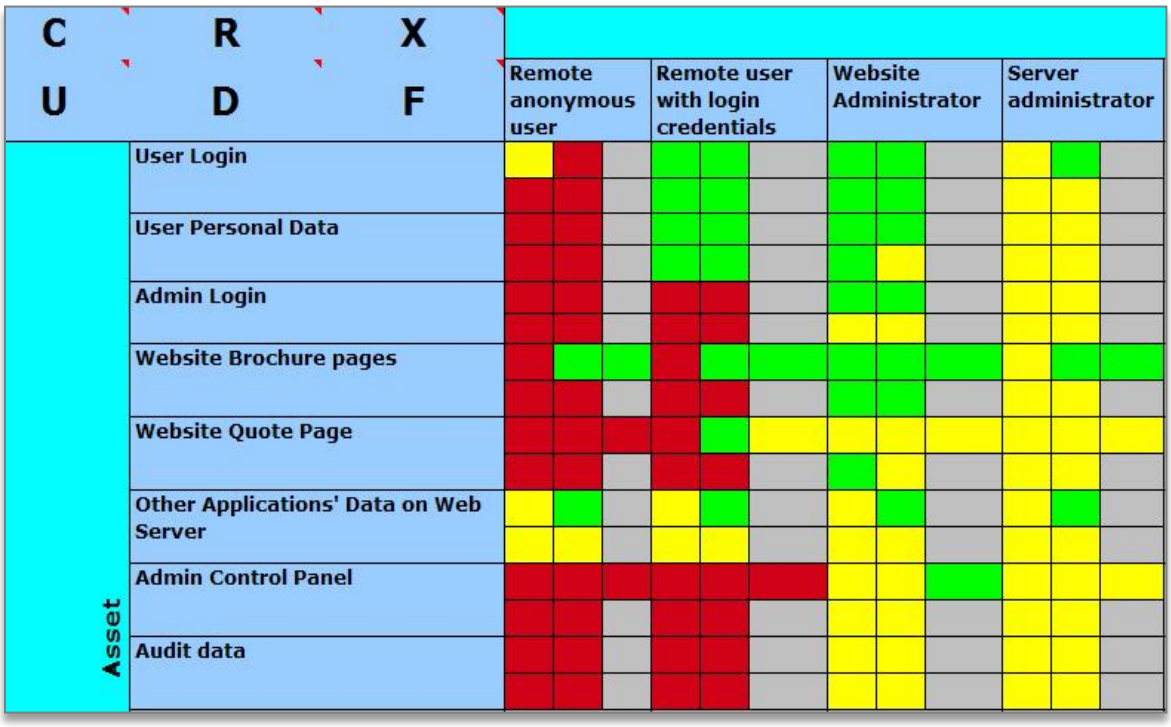


Figure 7 - CRUD privileges on each asset

As previously stated colour is important when displaying information in the TRIKE spreadsheet, each colour carries meaning (Table 6)

Colour	Description
Grey	This action does not apply to this asset, based on the asset's type in the Data Model
Red	(Never) The system should never let this actor take this action on this asset.
Yellow	(Conditionally) The system should let this actor take this action on this asset when certain conditions (typically documented in the cell comment) are met.
Green	(Always) The system should always let this actor take this action on this asset.

Table 6 - TRIKE CRUD Colour Definition [37]

Working through each step in the process the system overview, actors, data model, assets and intended actions are defined. At this stage the tool automatically generates threats. The methodology at this point places security issues into one of two categories: Elevation of Privilege and Denial of Service. When the threat grid gets produced each threat status is by default set to 'Unknown'. The threat modelling team and business stakeholder(s) go through each threat and assign a priority rating: low, medium, high (Figure 8).

Figure 8 - Priorities Threats

Priority ratings colour scheme is identified in table 7. All threats of high priority get added to the systems security objectives, which in turn are used to control how analysis is pruned for the rest of the threat modelling process.

Colour	Description
Grey	This threat is not meaningful for this object, based on the object's type or the business rules.
Red	(High) The direct business effects of this threat, should an attacker manage to reach it, have immediate, high-value, negative consequences for the organization. The system should definitely defend against this threat.
Pink	(Medium) The direct business effects of this threat, should an attacker manage to reach it, have negative consequences for the organization.
Light Pink	(Low) The direct business effects of this threat are negligible.

Table 7 - Threat Rating Colour Scheme [37]

The security objectives are the system's ultimate security goals. These are the threats the TIRKE process intends to address.

The final major tabs in the TRIKE process identify use cases that are likely to affect the threats in the security objectives. HAZOP analysis is performed at this point. Information such as security analysis, design flaws and test cases are recorded here as well as details of implemented mitigations.

The TRIKE Threat Model is completely dependent on its tools, which are not fully developed. The methodology may not currently[3] be the most suitable choice for organisations introducing Threat Modelling. The lack of detailed examples and documentation of usage/implementation relegate it in comparison with STRIDE. For these reason STRIDE was chosen for the case study Threat Model of a legacy enterprise application.

4.5 TRIKE v STRIDE

TRIKE approaches Threat Modelling from a defensive position as opposed to that of an attacker's. The system being modelled is defined within the requirements of TRIKE. Threat types are auto-generated and the security objectives set based on these threats. This risk based approach provides an opportunity to view the system from a different security point of view.

[3] June 2013

As part of this research a practical comparison of TRIKE and STRIDE was performed. The results of this comparison is provided in Table 8.

COMPARISON[4]	TRIKE	STRIDE
Supporting Tools	Yes	Yes
Tool Dependency	Yes	No
Tool maturity	Under Development	Fully Developed
Analyses commenced	Requirements phase	Design phase
Automated threat generation	Yes	Yes
Threat Classification	Two	Six
Security viewpoint	Defensive	Offensive
Suggested Mitigations	No	Yes
Support for variations of attacks	Yes (HAZOP analysis)	No
Detailed support Documentation	No	Yes
Ease of Implementation	Moderate	Easy/Moderate
Architecture Validation	No	Yes (DFD validation)
Generate Reports	No	Yes
Online tutorials	No	Yes
Sample Threat Models	No	Yes

Table 8 - STRIDE v TRIKE Comparison

TRIKE places threats into two categories (Elevation of Privilege and Denial of Service) as opposed to the six STRIDE classifications. Using the information generated by the spreadsheet to identify detailed threats however still comes down to the knowledge base and skill of the Threat Modelling team. The STRIDE classification provides suggestions of mitigation techniques based on the threat type. This greatly simplifies the process of identifying and implementing mitigations.

[4] June 2013

TRIKE uses HAZOP analysis. This allows for multiple variations of a process to be documented in a structure manner, similar to threat trees in concept but much easier to implement and maintain.

The gathering of information about the system being modelled is in a more systematic manner compared to STRIDE. It asks more specific, detailed questions about the system, its components, actors, intended actions and how they all connect. Each step is acquiring data that is used in the next step (tab).

One area of concern for TRIKE is the over reliance on colour in its spreadsheet to represent different information. Visually impaired people may struggle to distinguish between colours. This was confirmed by a colleague who is colour-blind. When presented the TRIKE spreadsheet this colleague was unable to distinguish between colours, therefore struggled to use the spreadsheet in its intended way.

STRIDE can be performed both manually and with the aid of tools compared to TRIKE which is heavily reliant on its tools. The methodology's theory is hard-coded into these tools which are still under development. The spreadsheet itself is not yet fully implemented and contains placeholders for future features. Documentation to assist TRIKE implementation is also sparse and no detailed sample TRIKE based Threat Models have been documented. This lack of information made the spreadsheet difficult to implement.

4.6 The use of Tools in Enterprise Applications

Threat Modelling can be very time consuming. To assist the process, a number of supporting tools have been developed as discussed in Chapters 3 and 4 of this research. In the opinion of this author, enterprises consider not only the methodology but also the tools available to support that methodology. Enterprises consider factors such as cost, accessibility, training, available documentation and tuition? Although the TRIKE spreadsheet is open source, the learning curve in applying the tool is greater than that of the Threat Modelling tool. The Threat Modelling tool is free from Microsoft but is dependent on Visio, which does require a licence. This tool produces faster results than TRIKES spreadsheet and groups the Threat Modelling data

in a more user-friendly, manner. The Threat Modelling tool is superior in maintaining a Threat Model. Its provides DFDs validation, analyses, presents large volumes of data in user-friendly UI, and thus more suitable to enterprise-sized applications

4.7 Conclusion

The TRIKE methodology is a viable option for organisations considering different Threat Model processes. TRIKE's weaknesses are STRIDE's strengths. STRIDE is fully developed with a large volume of resources including Microsoft's Threat Modelling Tool. After considering both approaches, the STRIDE based approach was chosen for the Threat Model case study of a legacy enterprise system.

CHAPTER 5.

THREAT MODELLING AN EXISTING LEGACY ENTERPRISE SYSTEM

Microsoft's SDL introduces Threat Modelling during a systems design phase, while TRIKE allows for a Threat Model to be started during the requirements phase, potentially identifying vulnerabilities in the systems requirements. Threat Modelling can be performed on existing projects. Information and examples of issues with relation to Threat Modelling existing/legacy systems is limited to a few sentences within documentation covered by this research. It is the opinion of this author that Threat Modelling of a legacy system would result in unique challenges compared to those faced by a new project.

The term legacy system refers to old/historical technology, application or computer system that remains in use. There are many reasons why such systems are maintained:

- The system is large and complex. The cost of a redesign or redevelopment exceeds the benefits of replacing the existing system.
- The system is stable and the owners see no reason to replace it.
- The system is in a state of Vendor lock in, resulting in considerable cost to change.
- The demand for constant availability means the system cannot be taken out of service.
- The application is poorly documented or the original documentation has been lost. In large organisations the original developers may no longer be with the company resulting in limited knowledge of how the system works.

Threat Modelling of a legacy project may provide unique challenges. Documentation that contains original system designs, code overviews, security assumptions and reports detailing original threats and mitigations may be limited or non-existent depending how old the application is. In such scenarios a lot of reverse engineering would be required while analysing

the system. Legacy projects that are still supported and have evolved over a period of time may not have a clear architectural outline. These projects may have had a large number of developers working on different parts of the system with varying degrees of proficiency in secure coding techniques. These types of applications may also be vulnerable to new threats as a result of new technologies such as the emergence of mobile technologies.

The implementation of a Threat Model on an existing legacy system would benefit the project by potentially identifying unknown threats. New risks have evolved with the emergence of new technologies. Legacy systems may not incorporate mitigations for these potential vulnerabilities. Organisations need to assess the risk to a business if an attack was successful and that information made public.

Although there might not be written documentation of the system, in-depth information may exist in the knowledge base of the people who currently support the project. The Threat Modelling process would involve documenting this knowledge. The Threat Modelling Tool has refined the STRIDE process to Diagram first as outlined in Chapter 4. For the case study presented in this research the original steps of collecting data about the system before implementing STRIDE were performed. These steps were used to extract and document the current knowledge of the legacy system from the existing development team.

Once the initial Threat Model is complete the project is open to the adoption of new development methodologies such as Agile SDL. The discussion of Agile and legacy projects is beyond the scope of this research however a key requirement to Agile SDL is the development of a baseline Threat Model of the system.

When researching software security and the topic of Threat Modelling many examples and case studies were found to be based on new applications with Threat Modelling starting early in the development lifecycle. This is not possible for older systems. However the process can still be greatly beneficial to existing applications. For this reason it was decided that the case study for this research would be based on the implementation of a Threat Model on an existing legacy system. The need for this research has been established in the previous sections.

CHAPTER 6.

CASE STUDY REPORT

As part of this research, a case study Threat Model was performed on an existing enterprise application. A company located in Ireland agreed to have its systems Threat Modelled with the understanding that the identification of the company, its application and any security issues identified during the study would remained confidential. Agreements to this effect were signed prior to commencing this Threat Model. Previous security audits had been performed on the application. No Threat Model had previously been performed. Within this report details such as the name of the application, its components, the companies name and the names of Industry participants have been changed to maintain anonymity. The systems architecture and security vulnerabilities identified have been maintained as close to the original system as permitted. If you have any questions please forward to supervising lecturer Ruth Lennon, Letterkenny Institute of Technology, Co. Donegal, Ireland. Email: ruth.lennon@lyit.ie

Technologies used to perform case study Threat Model included:

- STRIDE
- Microsoft Threat Modelling Tool
- Microsoft Visio
- DREAD

APP X is a legacy project over 15 years in existence. The system has continuously evolved to meet the changing business requirements of Company X Industry. This Threat Model focuses on the baseline application

6.1 Threat Modelling Information

The information in Table 9 provides an overview of the company and application. For the purposes of security the name of the company, application, version and other details have been sanitised. The original detailed Threat Model has been logged with the company and the

supervising lecturer. The enterprise application in this case is renamed APP X throughout the remainder of this document. Version X is the annotation used to represent the Enterprise Application that has undergone a number of revisions over the past number of years. The name of the industrial reviewer is also hidden for similar purposes.

Product	APP X
Milestone	Version: X
Owner	Company X
Participant	Michael McGrath
Reviewer(s)	Mr N. (Lead Technical Developer on APP X) Ruth Lennon (LYIT – Academic Supervisor)
Location	Application Base-line version, In-house staging environment.
Summary	APP X core processes include: • Management of business logic component • Allocation of resources • Distribution of Information • Supply management and planning information APP X is deployed on a private network (Figure 1) with multiple clients connecting to a central database. The solutions goal is to provide easy-to-use modules for operational planning, optimization of resources and equipment, and proactive control over component, giving the business flexibility needed to plan and improve business results.
History	APP X is a legacy project over fifteen years in existence. The architecture over that time has developed with both a 2 tier (Client/Database) and 3 tier (Client/Server/Database) approach used for different sections of the system. Multiple applications have also been developed in parallel to APP X using the same central database.

Table 9 - Threat Modelling Information

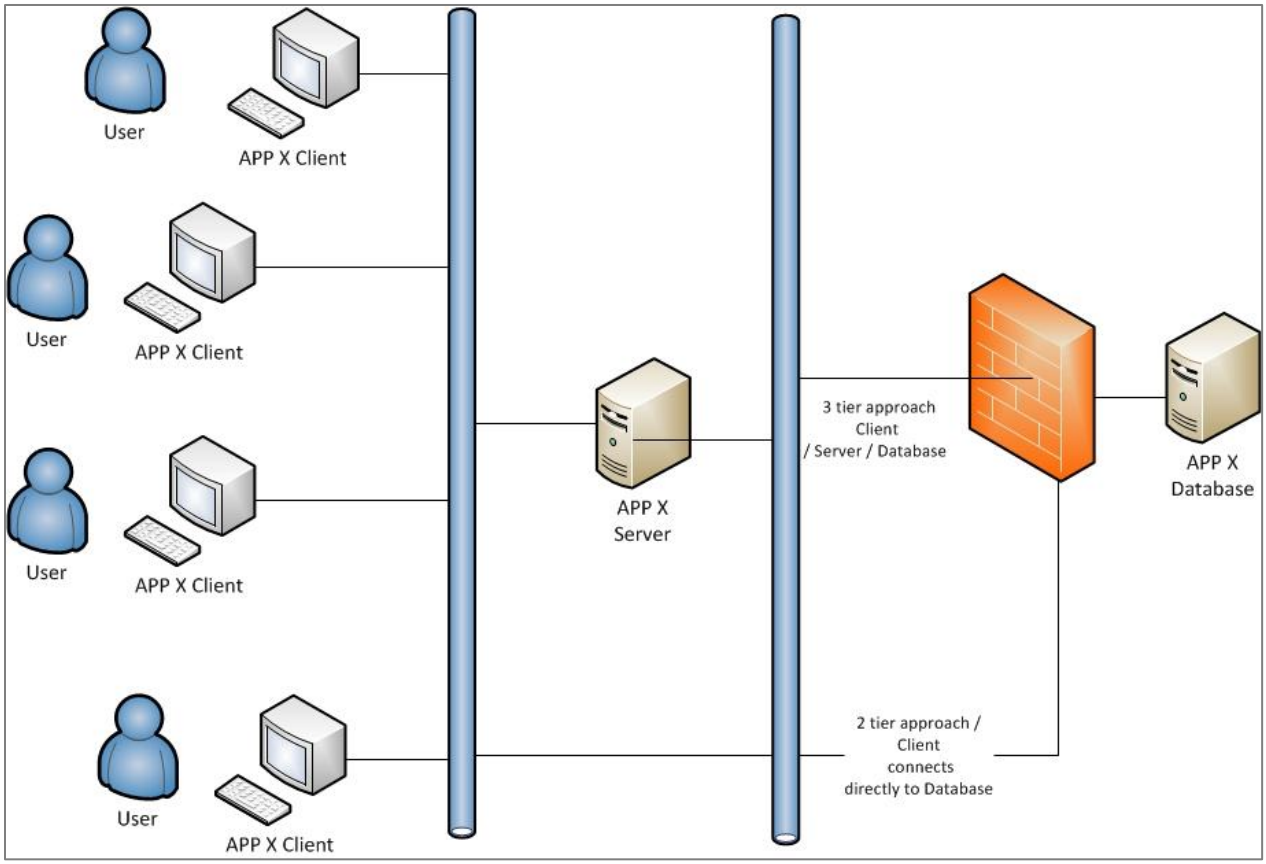


Figure 9 – APP X network Deployment

The threat modelling process performs several steps to document the application, its users, expected usage, dependencies, security notes and assumptions. This information was gathered from the knowledge base of the current development team.

6.2 Use Scenarios

List the known use scenarios (Table 10) for the application. This table provides information about the expected use of the application. Using or deploying the application in a way that violates a use scenario can impact the security of the application

ID	USE SCENARIOS DESCRIPTION
1	The APP X Database is installed on a database server that has been secured to current industry guidelines. Current security patches for the database server must be maintained. Physical security of the database server is site specific.
2	The APP X Server is installed on a server that has been secured to current industry guidelines. Current security patches for server must be maintained. Physical security of the application's server is site specific.
3	The APP X Client is installed on each work station.
4	Communication between APP X Client, APP X Server and the APP X Database is conducted over a private network within the customers building. Physical Security of network is site specific.
5	APP X requires user's login to access system. No access is permitted to anonymous users.
6	APP X behaves as an interface. Customer defined interfaces which update table's (Live data)
7	The availability of APP X functionality is based on user rights. Different functionality can be specified to different users/groups. Some functions are restricted to super users.
8	User authentication is performed by the APP X Client
9	APP X has three interfaces • User Interface 1 • User Interface 2 • User Interface 3
10	The application is designed for multiple user's to be logged on and using the system at any one time
11	The System has multiple external interfaces that interact with the database: External (Ex) Interface 1, Ex. Interface 2, Ex. Interface 3, Ex. Interface 4, Web Interface and Ex. Interface 5.
12	The application's Database send/receives data to/from 'Ex. Interface 1'
13	The application's Database send/receives data to/from the 'Web Interface'.
14	The application's Database send data to the 'Ex. Interface 2'
15	The application's Database send/receives data to/from 'Ex. Interface 3'
16	The application's Database send/receives data to/from 'Ex. Interface 4'
17	The application's Database receives data from the 'Ex. Interface 5'

Table 10 - Use Scenarios

6.3 External Dependencies

List of external dependencies the application has on other components or products that can impact security. These dependencies are assumptions made about the usage or behaviour of those components or products.

EX. DEPENDENCY	DESCRIPTION and ASSUMPTION
1	The applications Database runs on SQL Server 2008 R2 on a dedicated database server within the customers' network. The database server is run on Windows OS
2	Operating Systems: Win XP 32/64, Windows 7 32/64
3	APP X Server can be installed on the same server as the Database or a dedicated server within the customer's network. Site specific
4	The security of Database is dependent on the security of the DB server it is installed on
5	The security of APP X server is dependent on the security of the server it is installed on
6	The security of APP X is dependent on the security of each machine the Application client is installed
7	APP X depends on the security of the customer's network. If this network is compromised, sensitive data could be viewed, or direct attacks on the database or APP X Server could be attempted
8	The Security of APP X is dependent on the security of each external Interface that interacts with the APP X Database
9	Anti-virus software – site specific
10	Firewalls – site specific

Table 11 - External Dependencies

6.4 External Security Notes

External Security notes – threats or other information that an application user should be aware of to prevent possible vulnerabilities. These notes can include features that, if used correctly, could cause security problems for application users.

ID	NOTE
1	Session management does not provide inactivity log out. The system will not time out if left idle.

Table 12 - External Security Notes

6.5 Internal Security Notes

Internal Security notes – contain security related information relevant only to someone reading the Threat Model. These notes can also be used to explain choices and design decisions that impact the products security but were made due to overriding business needs or due to legacy nature of the system

ID	NOTE
1	Login authentication communication with the applications Database is not encrypted. Potential for eavesdropping
2	Part of APP X Client which is a specific section of the applications source code that was developed in a 2 tier approach. This part of the application requires direct access to the systems Database. For this reason firewall ports remain open
3	User Authentication is performed by the APP X client
4	No mutual authentication is performed by the client / server. Connections are made by a client / server handshake protocol.
3	The system requires the APP X Client details in the Client Server are configured. When the application is run it displays in plain text the username and password used to establish the connection to the database server. The Consequence of having these configurations in plain text is that privileged accounts (SA with Sysadmin privilege) and password is relatively accessible, readable and usable. This means and represents a high risk in the case someone besides the custodian of the account gets the password

Table 13 - Internal Security Notes

6.6 Assets

Assets describe the data or functionality that the component needs to protect. This table also lists the minimum access category (trust level) that should be allowed to access the resource.

ASSET ID	Name	Description
1	User Login data	User Credentials: username / password
2	System Login data	System super admin credentials: username / password
3	Database data	All data that is stored with the systems Database
4	Availability of APP X	If APP X goes down, customers cannot manage their business components and allocation of resources.
5	Login Session	The session associated with a logged in user
6	Access to Database	The ability to interact with the database that stores user credentials, business components, billing and other data of value with Database
7	Up-to-date data	If APP X speed slows this can affect operations as the application relies on accurate up-to-date data
8	Accuracy of data	APP X requires accurate data. Inaccurate data can have severe impact on the businesses operations
9	Audit data	Adversaries might try to attack the system without being logged or audited
	Access to the APP X Server	Direct access to the server could result in attacks against the server or use of the server to access the Database

Table 14 - APP X Assets

6.7 Entry / Exit Points

Entry/Exit points describe the interface through which external entities can interact with the component, either by direct interaction or indirectly supplying it with data.

ID	Name	Description	Trust Level
1	APP X Client	User input through APP X clients: • Multiple Clients can be connected to APP X Server	APP X User APP X Admin
2	Patch Management files	Updates files sent to APP X	System Admin
3	Database Logging files	Database Log files written by APP X	System Admin
4	Client Logging files	Client Log files written by APP X	System Admin
5	APP X Client	APP X client interactions that directly communicate with the Database	APP X User APP X Admin
6	Ex. Interface 5	Data sent from Ex. Interface 5	-
7	Web Interface	Data sent to/from Web Interface	-
8	Ex. Interface 1	Data sent to/from Ex. Interface 1	-
9	Ex. Interface 2	Data sent to Ex. Interface 2	-
10	Ex. Interface 3	Data sent to/from Ex. Interface 2	-
11	Ex. Interface 4	Data sent to/from Ex. Interface 4	-

Table 15 - Entry / Exit Points

6.8 Assumptions

Assumptions are notes taken before/during the threat modelling process

ID	NOTE
1	The scope of this Threat Model includes the APP X Client, Server and Database
2	External Interfaces are deemed separate applications which require individual Threat Models.
3	The application is deployed on a secure network. A Potential adversary will most likely be a user with some level of access to the network the application resides on.
	2 and 3 tier architecture implemented due to legacy nature of project

Table 16 - Assumptions

6.9 Data Flow Diagrams

The APP X Threat Model provides eight data flow diagrams (DFDs). Figure 10 shows the systems context diagram, and Figure 11 shows the Level 0 DFD. During the threat modelling process Level 1 diagrams (Figure 12, 13, 14 and 15) where created to identify the flow of data between individual components of the system. Additional Level 2 DFDs were also created to demonstrate the flow of data during a user login and a client / server communication, as these areas where identified during the threat modelling process as possible entry points for potential adversaries.

Note Data Flow Boundaries:

- Red dash Line – Data crossing a Trust Boundary
- Blue dashed Line – Data crossing a Machine Boundary

6.9.1 APP X – DFD Context Diagram

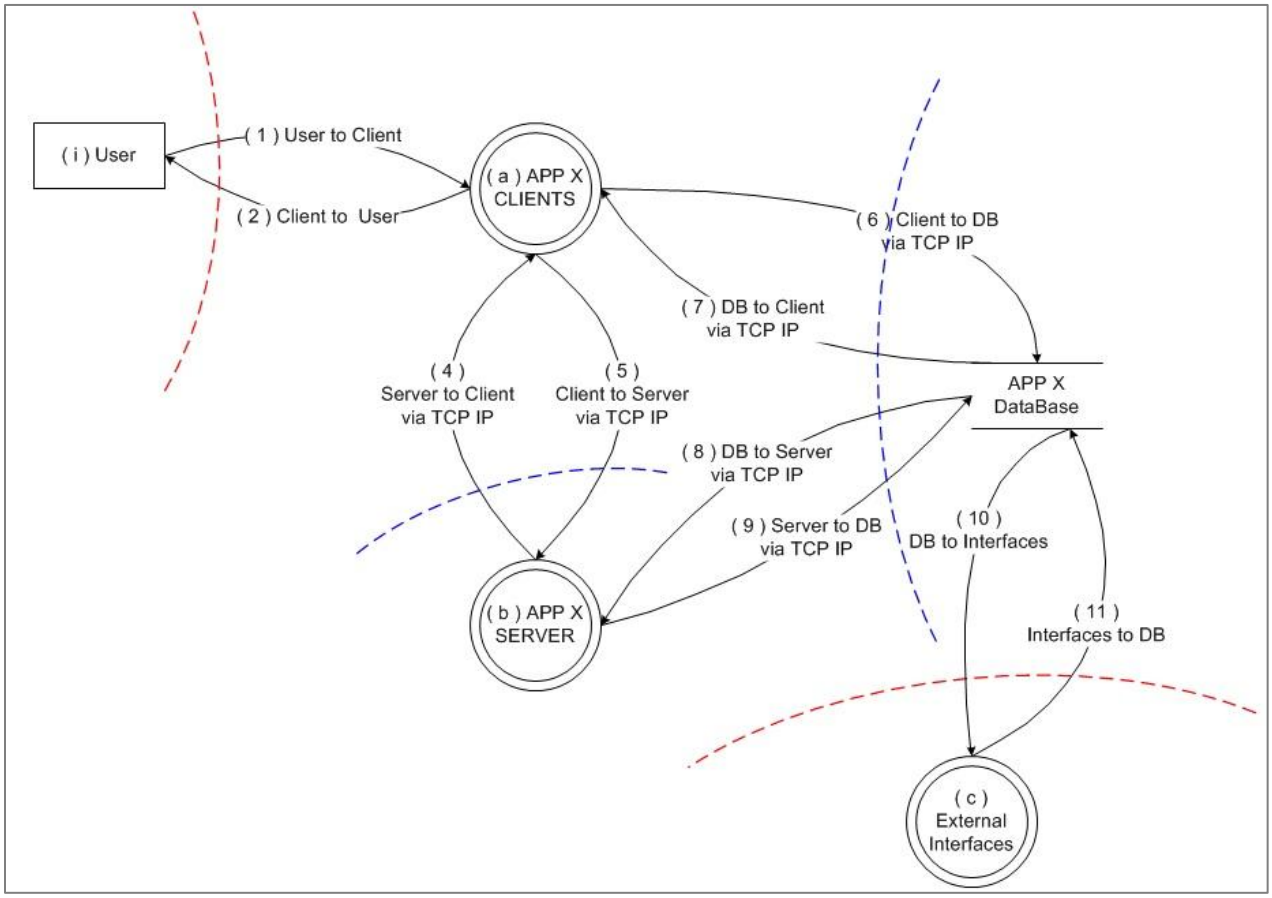


Figure 10 – APP X Context Diagram

APP X context diagram (Figure 10) provides a high level overview of the system. The diagram shows one external interact or (i) User, which has two way data flow between the APP X Client. This data flow crosses a trust boundary (depicted by a red dashed line) as the user and it's interactions with the client is non-trusted flow of data.

The APP X Client(s) are depicted by the multi-process '(a) APP X CLIENTS' which has a two dataflow with both the APP X Server depicted by the multi process '(B) APP X SERVER' and the Database. Dataflow to/from '(a) APP X CLIENTS' to both '(B) APP X SERVER' and Database crosses machine boundaries (depicted by a blue dashed line), which identifies data moving from one machine to another.

The APP X Server also shows a two way data flow with the Database which crosses a Machine Boundary. As previously stated the Database has a two way dataflow with both the APP X

Server and Clients. The DFD also shows two dataflow directly between the database and external interfaces depicted by multi processes '(c) Interfaces'.

6.9.2 APP X (DFD Level 0)

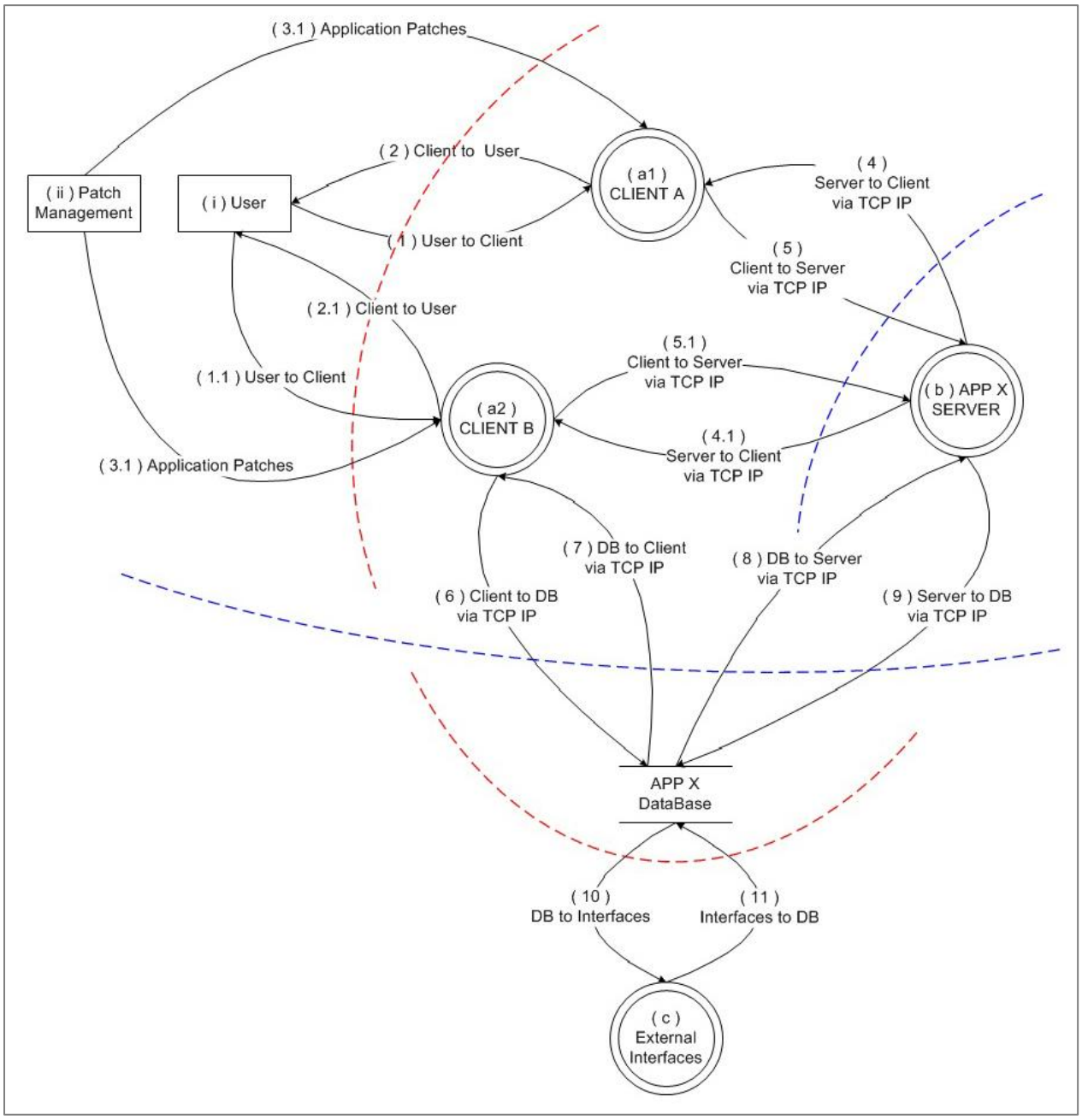


Figure 11 – APP X DFD Level 0

APP X DFD Level 0 (Figure 11) provides a more detailed view the system. The multi-process '(a) APP X CLIENTS' identified in the context diagram (Figure 10) is defined in further detail. '(a1)

Client A' identifies aspects of APP X client which use a 3 tier approach (Client, Server, Database), with all interactions with the database performed through the APP X Server.

'(a2) Client B' identifies aspects of the APP X client which uses a 2 tier approach (Client, Database), with interactions directly with the Database. The 2 and 3 tier approaches represent different aspects of the Client that were developed at different times during the evolution of this system. A security concern was obvious from this DFD and before STRIDE was implemented. The direct communication by the 'Client B' and Database.

An additional external Interactor (ii) Patch Management has been added at this level

6.9.3 APP X Server (DFD Level 1)

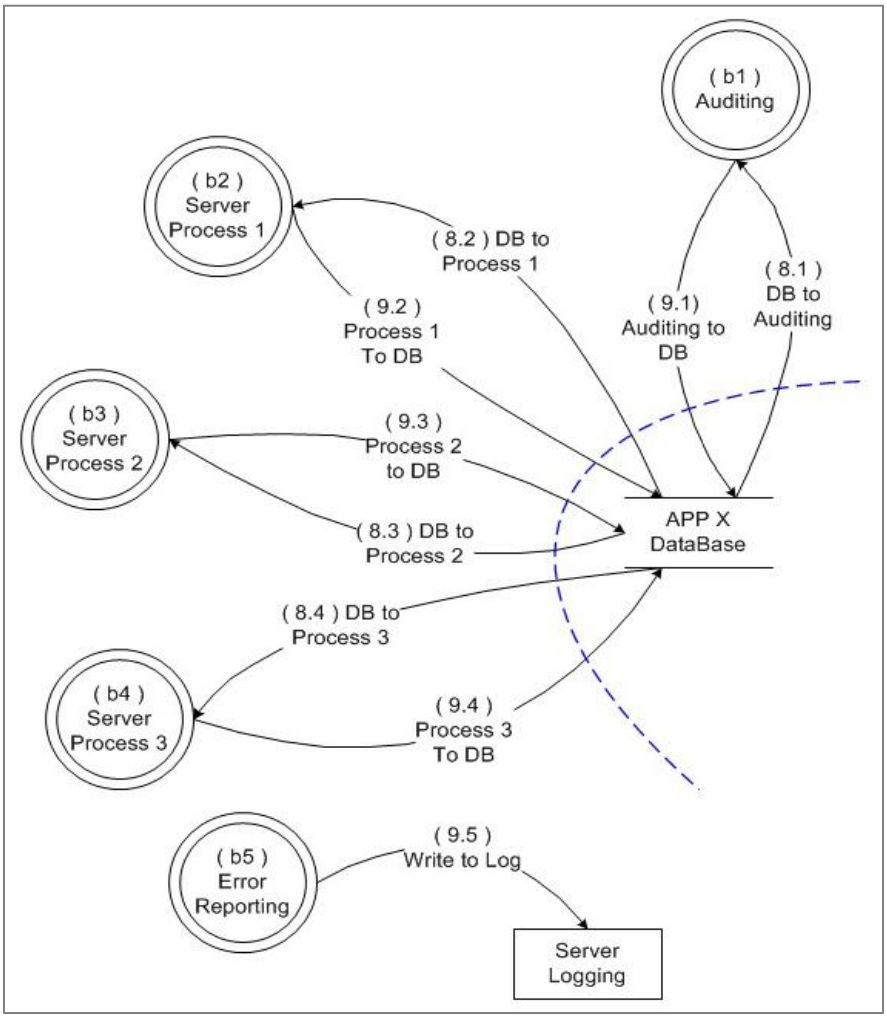


Figure 12 - APP X Server (DFD Level 1)

APP X Server Level 1 DFD (Figure 12) shows a high level overview of the internal processes within the Server and flow of data between these processes the systems Database. The key processes include (b1) Auditing, (b2) DB Operations, (b3) Server Process 1, (b4) Server Process 2. The server error logging is also depicted by multi process (b5) Error Reporting. Logs are recorded and located on the same server that the APP X server is installed.

6.9.4 APP X Client B (DFD Level 1)

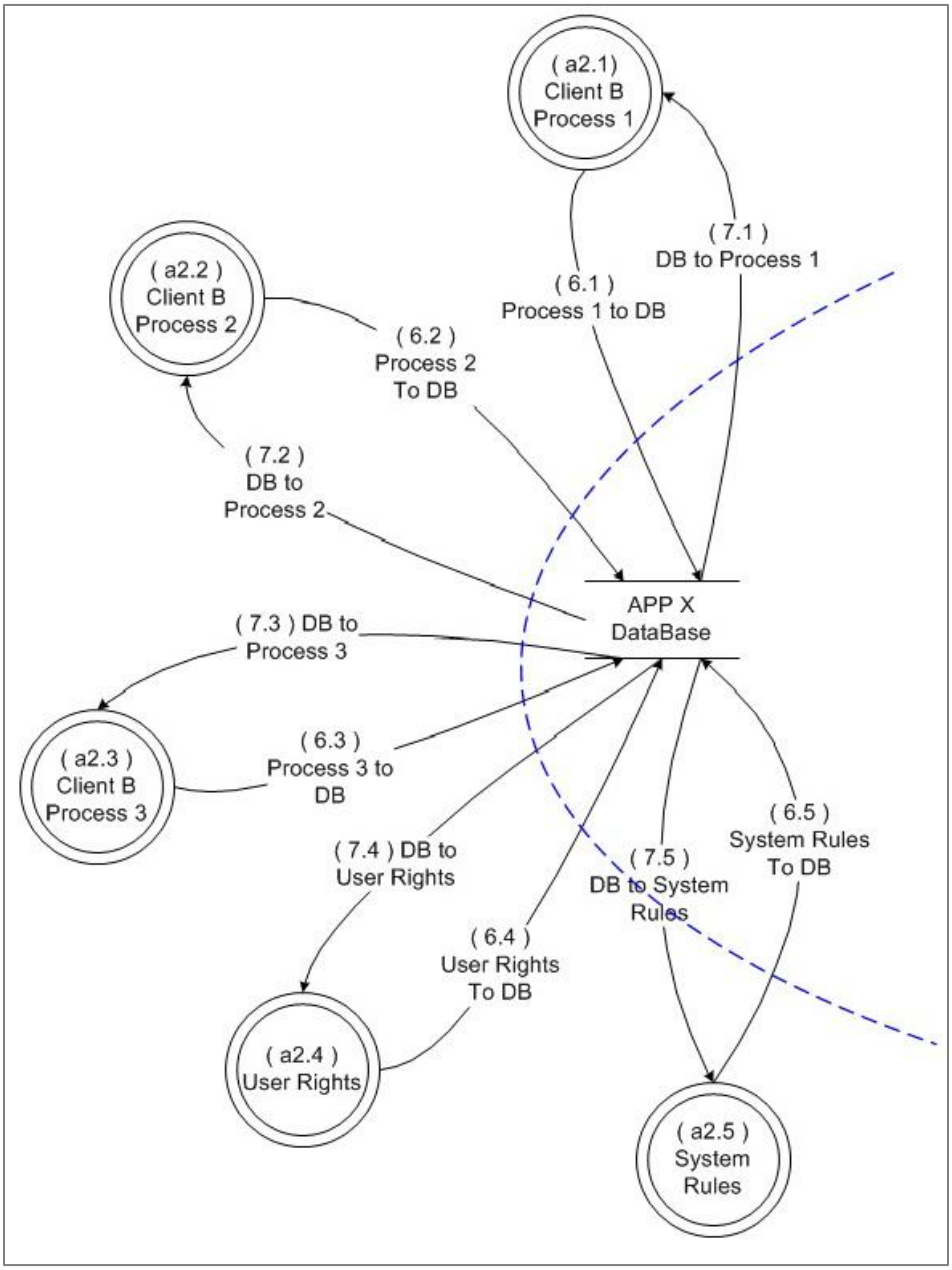


Figure 13 – APP X Client B - Level 1 DFD

Client B Level 1 DFD (Figure 13) shows a high level overview of the internal processes within the APP X Client B and flow of data between these processes the systems Database. The key processes include (a2.1) Billing, (a2.2) Archiving, (a2.3) Client B process 1, (a2.4) User Rights, (a2.5) System Rules.

6.9.5 APP X Client A (DFD Level 1)

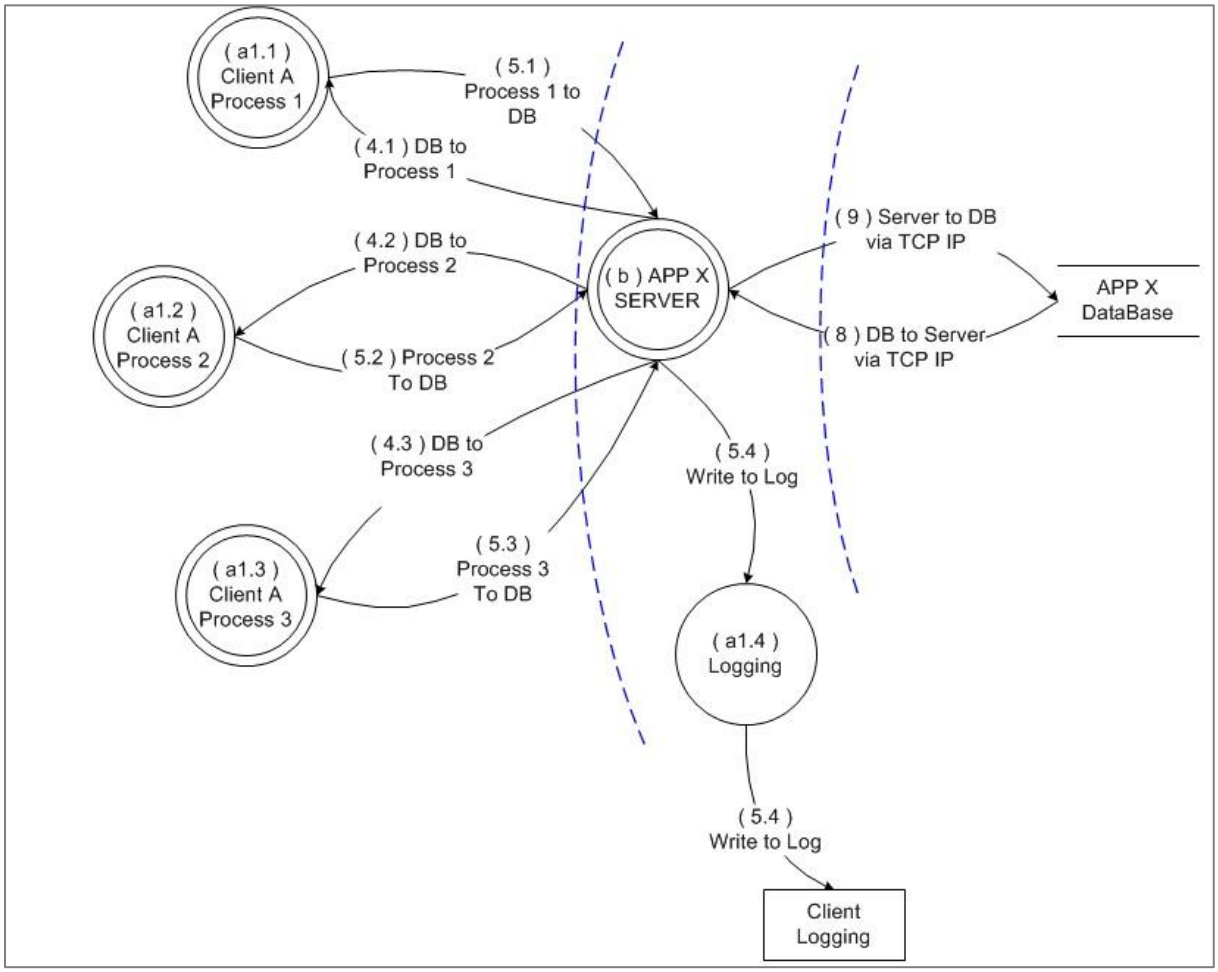


Figure 14 – APP X Client Level 1 DFD

Client A Level 1 DFD (Figure 14) shows a high level overview of the internal processes within the Client A and the flow of data between these processes the APP X Server. The key processes include (a1.1) Client A process 1, (a1.2) Client A process 2, (a1.3) Client A process 3.

The DFD also shows the client logging by the APP X Server depicted by multi process (a1.4) Logging. Logs are recorded on the same server that the APP X server is installed.

6.9.6 External Interfaces (DFD Level 1)

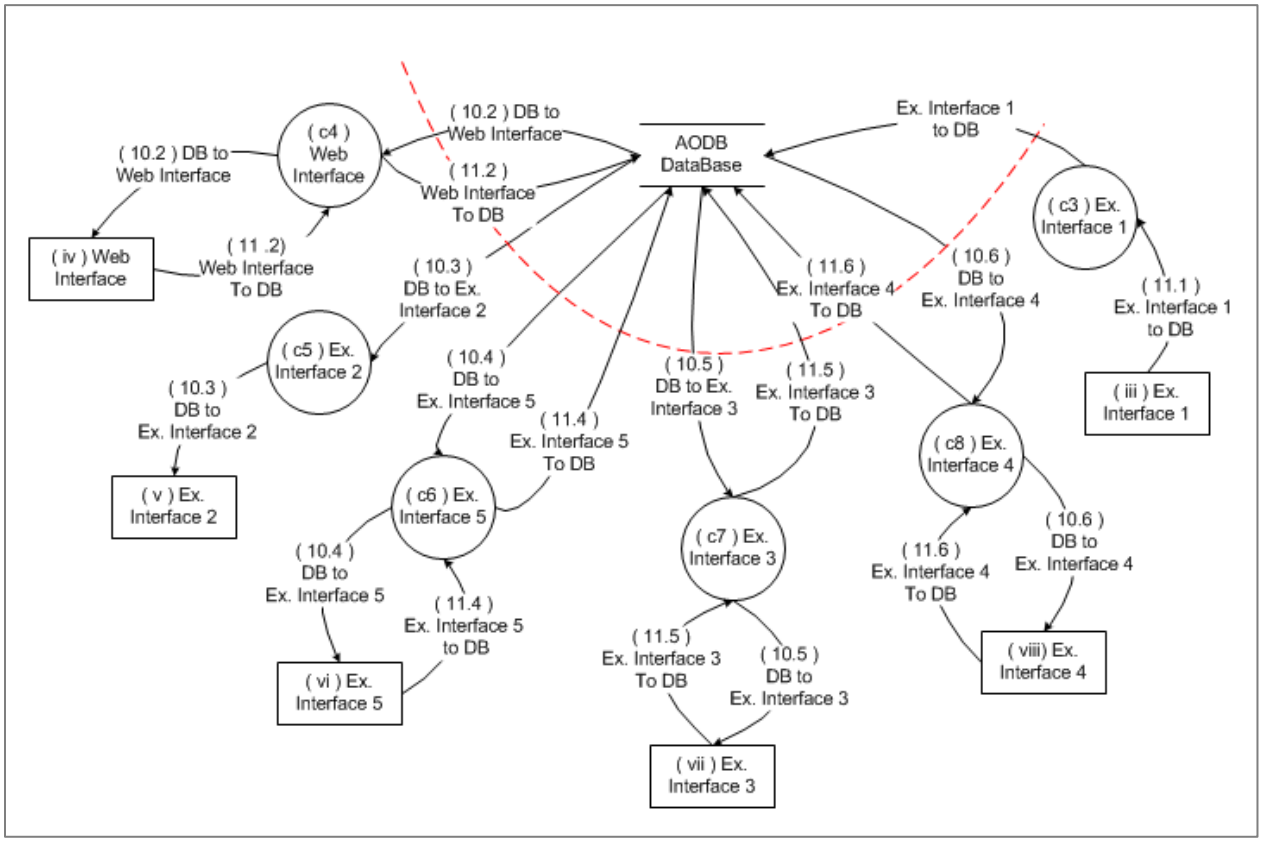


Figure 15 - External Interfaces DFD Level 1

External Interfaces Level 1 DFD (Figure 15) shows a high level overview of the external interfaces that interact with the applications Database and the flow of data to/from these external interactors. The external interactors identified include: (iii) Ex. Interface 1, (iv) Web Interface, (v) Ex. Interface 2, (vi) Ex. Interface 3, (vii) Ex. Interface 4, (viii) Ex. Interface 3

6.9.7 Client / Server Handshake Protocol (DFD Level 2)

Client Server Handshake Protocol Level 2 DFD (Figure 16) shows a high level overview of how communication between an APP X client and the APP X Server is established. This aspect of the system was investigated during the threat modelling process.

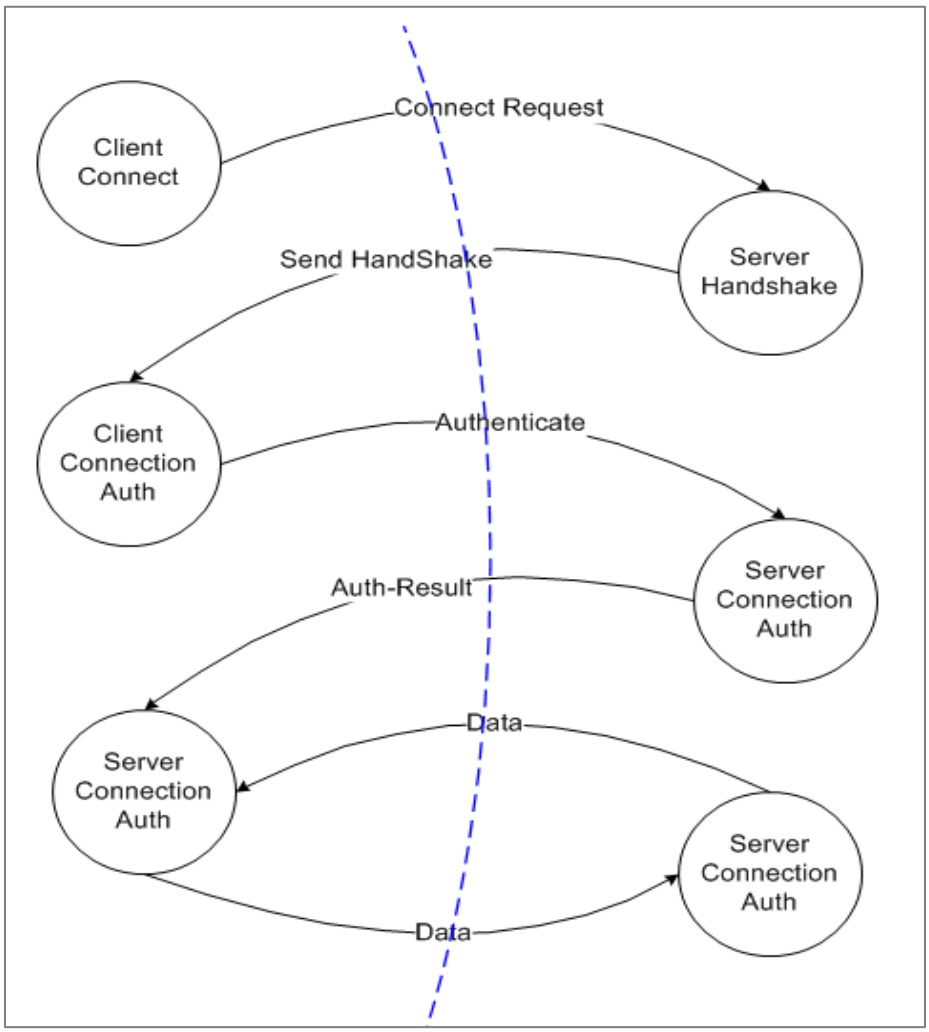


Figure 16 - - Client Server Handshake DFD Level 2

6.9.8 User Login (DFD Level 2)

The user login level 2 DFD (Figure 17) shows a high level overview login attempted is processed by the application. This aspect of the system was investigated during the threat modelling

process. The Client Login Process determines if the user can gain access to the system. Authentication is performed by the APP X Client.

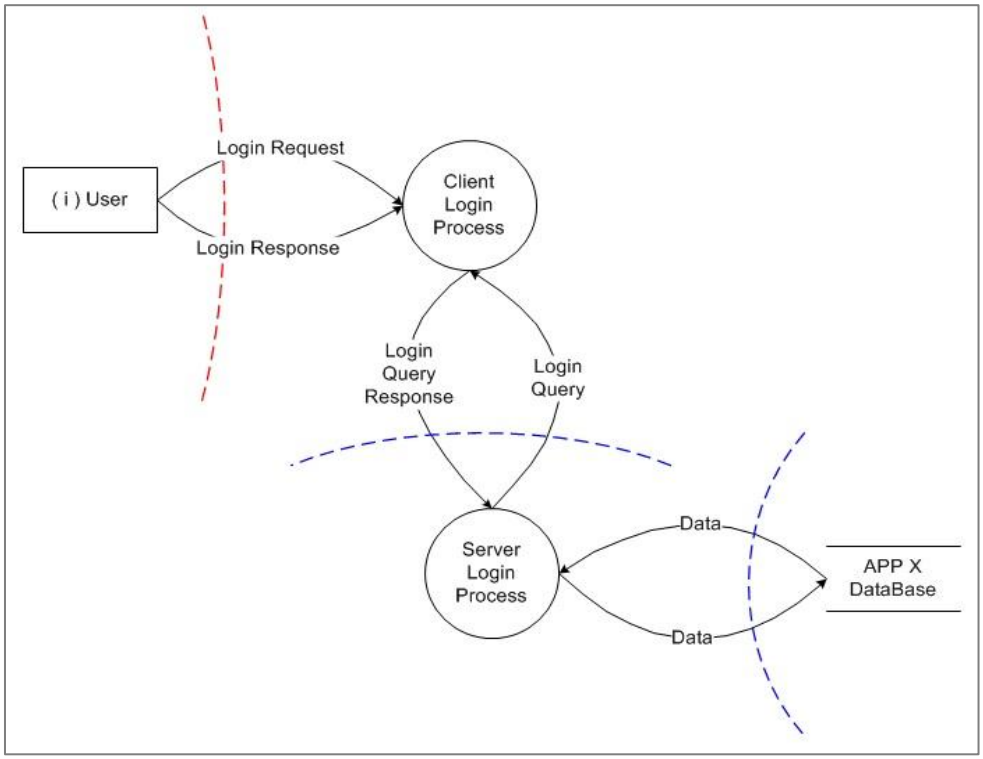


Figure 17 - - User Login DFD Level 2

6.10 STRIDE Process

Microsoft's Threat Modelling tool was used to assist the implementation of the STRIDE against each element in the projects DFD's. The tool provided an organised approach to assess each aspect of the system. This Threat Model started at a high level provided by APP X DFD level 0 (Figure 10) and from there assessed each individual element and data flow as identified by the systems DFDs.

The threat modelling tool automated the classification of threat types. For example Data Flow '(1) User to Client' based on STRIDE has three potential Threat Types: Tampering, Information Disclosure and Denial of service. As part of the process each threat type is evaluated against each element. The impact of the threat and solutions to those threats documented. The Threat Modelling Tool also allowed for additional management of threats including:

- Certify that there are no threats of this type
    - This allows for an individual threat against a specific element to be documented as non applicable and the reason for this decision to be document based on three groupings: within a trust boundary, mitigated elsewhere, accepted risk
- Flag a Threat as Finished
    - When a threat type has been evaluated and sufficient mitigations identified.

Enterprise scale applications may have a large number of potential threats generated by STRIDE. These features of the Threat Modelling Tool proved important in the overall management and of each threat.

STRIDE was implemented against each element in the system. Each potential threat generated by STRIDE was then assessed.

The STRIDE process identified areas of the system that have potential unmitigated threats. Table 17 is a list of elements that were identified as having insignificant or non mitigations in place.

THREAT TYPE	DFD ELEMENTS
Spoofing	**External entities:** • (i) User **Processes:** • (a1) Client A • (a2) Client B • (b) APP X Server • (c) Interfaces
Tampering	**Data flows:** • (11) Interfaces to DB • (2) Client to User • (2.1) Client to User • (4) Server to Client via TCP IP • (5) Client to Server via TCP IP **Data stores:** • Database **Processes:**

THREAT TYPE	DFD ELEMENTS
	• (b) APP X Server • (c) Interfaces
Repudiation	**External entities:** • (i) User **Data Store:** • Database **Processes:** • (b) APP X Server • (c) Interfaces
Information disclosure	**Data flows:** • (1) User to Client • (11) Interfaces to DB • (2) Client to User • (4) Server to Client via TCP IP • (5) Client to Server via TCP IP • (6) Client to DB via TCP IP **Data stores:** • Database **Processes:** • (a1) Client A • (a2) Client B • (b) APP X Server
Denial of service	**Data flows:** • (11) Interfaces to DB • (4) Server to Client via TCP IP • (5) Client to Server via TCP IP • (6) Client to DB via TCP IP • (7) DB to Client via TCP IP **Data stores:** • Database **Processes:** • (b) APP X Server • (c) Interfaces
Elevation of Privilege	**Processes:** • (b) APP X Server

Table 17 - Potential Threats Identified by STRIDE

Presented with the information from STRIDE which identified threats to the system and using this information along with the knowledge of the system acquired throughout the Threat Model a list of vulnerabilities was compiled outlining potential weaknesses in the system that could be exploited by an adversary.

6.11 Vulnerabilities

This section outlines the vulnerability identified during the Threat Model process. The unmitigated threats (Table 17) were used to compose a vulnerability listing.

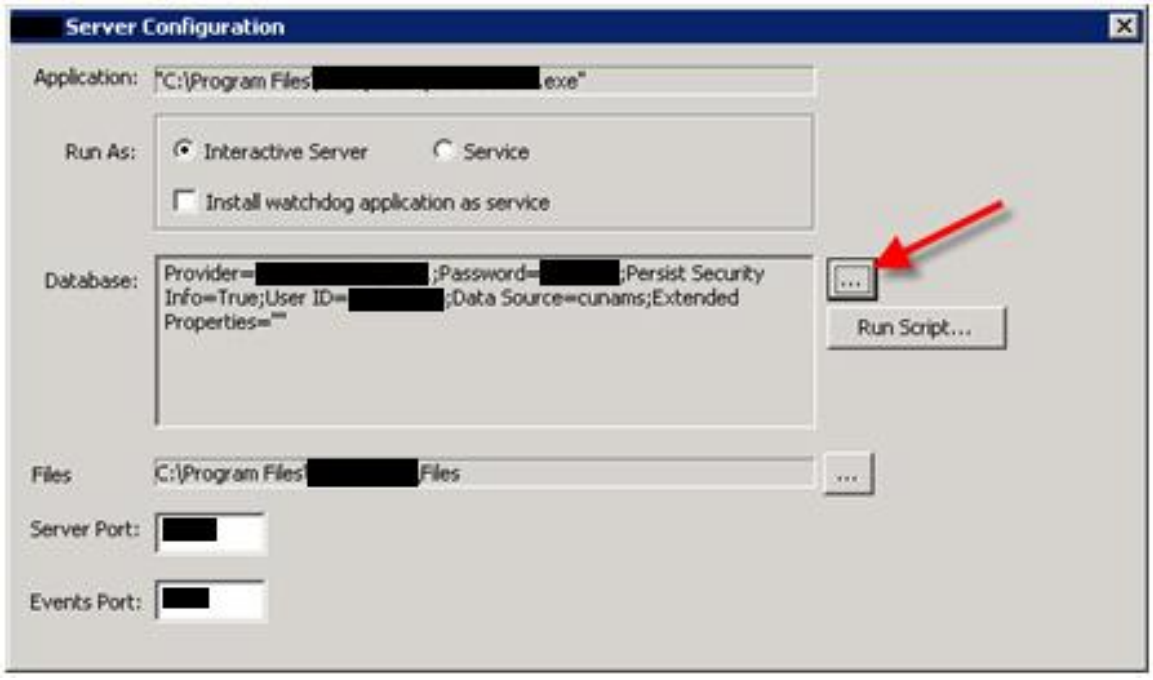


Figure 18- APP X Server Configuration

Name	Unauthorised Access to Database
ID	1
Description	The APP X database is located on a private network behind a firewall. Due to the 2 tier approach of parts of the systems architecture ports on the systems firewall remain open to allow communication between the Client A and the applications Database. An adversary could potentially gain access to the database through these ports. This vulnerability is aided by the internal security issue (5) Where the server configuration displays in plain text the username and password used to establish the connection to the database server (Figure 18). Access to this configuration is restricted to trusted Administrative users. Unencrypted traffic on the network could also assist a sophisticated attack against the Database by providing information on the internal structure of the database.
Mitigated?	Not currently mitigated. Legacy project, issue inherited. Solution requires refactoring of a large section of the system
STRIDE Classification	• Tampering • Information Disclosure • Denial of Service
Entry Points	No direct entry point through the application. Vulnerability potentially allows exploits of open ports on the firewall
Assets	• (3) Database data • (4) Availability of APP X • (6) Access to system Database

Table 18 - Unauthorised Access to Database

Name	Unauthorised Access to APP X Server
ID	2
Description	APP X uses the client / server handshake protocol to establish a connection between a client and server. Multiple clients connect to a single server. No Mutual Authentication exists between a client and the server. It is potentially possible to bypass the APP X client and connect directly to the server. An adversary requires only the IP address of the server, access to the network Server and Event Ports. APP X User authentication is performed by the client. Therefore if the client is bypassed an adversary could potentially perform an attack either against the server or through the server against the database.
Mitigated?	Not currently mitigated
STRIDE Classification	• Spoofing • Tampering • Repudiation • Information Disclosure • Denial of Service • Elevation of Privilege
Entry Points	Bypassing '(1) APP X client' accessing system through the APP X Server
Assets	• (10) Access to the APP X Server • (3) Database • (4) Availability of APP X • (6) Access to Database

Table 19 - Unauthorised Access to APP X Server

Name	Disclosure of Login Information
ID	3
Description	If an adversary acquires the username and password of another APP X user. It would be possible to perform any task that user can perform. Network traffic is not encrypted.
Mitigated?	Not currently mitigated.
STRIDE Classification	• Information Disclosure • Elevation of Privilege
Entry Points	• (1) APP X Client
Assets	• (1) User Login data

Table 20 - Disclosure of Login Information

Name	Access without Auditing
ID	4
Description	Adversary compromises the system, but insufficient auditing is in place to know if an attack occurred and little evidence to indicate where the attack came from. With relation to Vulnerability 1 & 2. If these attacks where successful no auditing current in systems to identify source of security breach.
Mitigated?	Not currently mitigated.
STRIDE Classification	• Repudiation
Entry Points	• (1) APP X Client • Bypassing '(1) APP X client' accessing system through the APP X Server
Assets	• (9) Audit Data

Table 21 - Access without Auditing

Name	Un-vetted Interfaces
ID	5
Description	The APP X environment incorporates a number of external interfaces. Un-vetted Interface interactions with the database exist. Similar to access without auditing however this is specific to the interactions of external Interfaces with the Database The APP X logs can identify APP X interactions with the Database. All other calls to the database are not logged.
Mitigated?	Not currently mitigated.
STRIDE Classification	• Repudiation
Entry Points	• (6) Ex. Interface 1 • (7) Web Interface • (8) Ex. Interface 2 • (9) Ex. Interface 5 • (10) Ex. Interface 3 • (11) Ex. Interface 4
Assets	• (9) Audit Data

Table 22 - Un-vetted Interfaces

Name	Network traffic Interception
ID	6
Description	Network Traffic is not currently encrypted. An insider threat could monitor network traffic
Mitigated?	Not currently mitigated.
STRIDE Classification	• Spoofing • Tampering • Elevation of Privilege
Entry Points	NA
Assets	• (1) User Login Data • (2) System Admin Login Data

Table 23 - Network traffic Interception

6.11 DREAD Scoring

For each vulnerability listed after the STRIDE activity, a score of 1 (lowest) to 3 (highest) as to the severity/impact of that vulnerability in each of the dimensions of DREAD was awarded. Each of these component scores are combined to give a total DREAD score for each vulnerability.

Reminder of DREAD acronym:

Damage Potential, Reproducibility, Exploitability, Affected users, Discoverability.

The Risk Rating was assessed as High/Medium/Low according to the score:

- 5-8 = Low Risk
- 9-12 = Medium Risk
- 13-15 = High Risk

VULNERABILITY	D	R	E	A	D	TOTAL	RISK RATING
1. Unauthorised direct access to Database	3	2	3	3	2	13	H
2. Unauthorised direct access to Server	3	1	1	3	1	9	M
3. Disclosure of Login Information	3	1	1	2	1	8	L
4. Access without Auditing	3	2	2	3	2	12	M
5. Un-vetted Interfaces	3	2	2	3	2	12	M
6. Network traffic Interception	3	1	2	3	3	10	M

Table 24 - DREAD Score sheet

CHAPTER 7.

VERIFICATION

The Threat Modelling process produced a list of vulnerabilities identified by STRIDE and assessed further by DREAD. The case study enterprise application was used daily by a large number of Company X customers. The legacy status of the project and the nature of some of the vulnerabilities identified could result in possible high costs to implement mitigations necessary to prevent these vulnerabilities from exploitation by potential adversaries. In the knowledge that the finding's of this research would have an impact on a real-world company and potential implications for their customers, each vulnerability where applicable was investigated further and methods to exploit these vulnerabilities identified. Company X was presented with the findings of this Threat Model and a demonstration of how some of those vulnerabilities could be exploited was also provided. It was hoped the information presented to Company X would assist with the decision making necessary to move forward with their application.

This section documents the penetration testing performed to identifying potential methods that could be used by an attacker to exploit the security vulnerabilities in APP X. Each test was prepared in advance. Company X provided access to the live production system. With full permission from the company, ethical hacking was performed on APP X to verify the Threat Models results. Each test was from the viewpoint of an adversary and how they could potentially carry out such attacks.

7.1 Test: Unauthorised Direct Access to Database

Description

APP X was deployed on a customer's private network. The system consisted of three tier (client, server and database) architecture. During this research it was identified that a section of the legacy code operated in a two tier approach (Client / Database) with a section of the client communicating directly with the systems database. For this reason ports remained open on the

networks firewall to allow this section of the client software to communicate directly with the database.

This test aimed to demonstrate how a potential adversary could perform an attack against the database. The goal of the test was to connect to the database by exploiting the firewalls open ports and retrieve data once a connection was established (Figure 17). This test also demonstrated the ease with which such an attack could be successful.

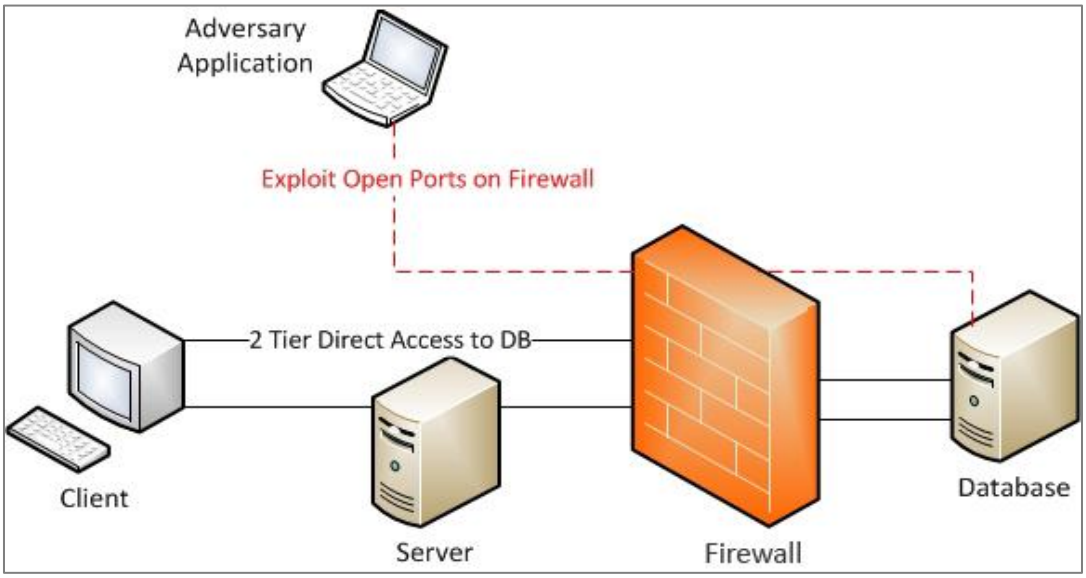


Figure 17 - Connection to Database via Open ports on private Network Firewall

The technologies used to connect to the database included XAMPP setup to run off a portable USB flash drive and PHP scripts to perform the connection and query of the database. XAMPP is an Apache web server distribution containing MySQL, PHP and Perl [38]. This setup could allow a potential attacker to prepare scripts in advance and perform the attack quickly when access to the network had been established.

Assumptions

- Adversary has access to the private network.
- Adversary may be a 'trusted user'.
- Firewall ports are open to accommodate the two tier approach for a section of legacy system.

- Adversary must identify open ports.
- Adversary may use third party tools to identify open ports.
- Adversary must acquire Database credentials to gain access.
- An adversary may acquire Database credentials from plaintext server configuration as identified during Threat Model.
- Testing was carried out on a live production system.
- Detailed steps to acquire port & database credentials are beyond the scope of this test.

Test Steps

- A test database was created on the SQL Server.
- XAMPP lite was downloaded [39] and configured to run off a USB flash drive.
- Additional drivers (Listing 1) for MS SQL server were downloaded [40] and added to the XAMPP setup.

```
php_sqlsrv_53_nts_vc9.dll
php_pdo_sqlsrv_53_nts_vc9.dll
```

Listing 1 - Microsoft PHP SQL Server DLLs

- The php.ini file was configured to load the added DLL files [41].
- The XAMPP setup was run off a USB and a connection to a Microsoft SQL Server 2008 R2 was tested and confirmed.
- A PHP script was prepared (Appendix 5 – DB Script) to take input parameters including: Server, Database Name, Port Number, DB Type, Username, Password and a SQL Query. These parameters would be used to connect to the server, connect to database and then query the database. The script allowed for credentials and queries to be inputted and if successful return the queries results to a text box (Figure 18).

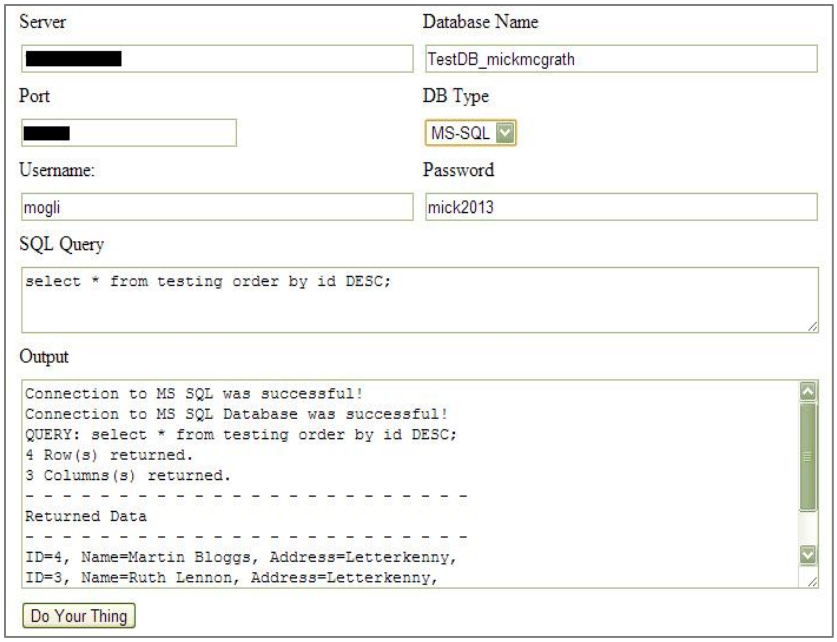


Figure 19 - Screen shot of input and output of DB connection

- This script combined with XAMPP can run from a USB. This setup provided a means to quickly connect and query a database. An attacker would need to focus only on acquiring port number(s) and database server credentials.
- An adversary may use open source tools such as Superscan 4.0 [42] to perform IP and port sweeps identifying live hosts and opened ports. This tool was tested and run from a USB.
- Combining XAMPP, the PHP script and open source scan tools, an attack against the database was simulated within the live production system.

Results of test run on live production system

- XAMPP successfully ran from a USB flash drive.
- PHP script successfully ran on XAMPP.
- Providing the PHP connection script with the correct port number, server IP and database credentials it was possible to connect to the database through open ports on the firewall and return the results of a query run against the database.

The aim of this test was to demonstrate a method to connect and query APP X database by taking advantage of open ports on the systems firewall. This test was completed successfully.

From here an adversary could commence retrieving, altering or deleting data in the applications database.

STRIDE Classification

Successful implementation of test scenario could result in:

- Tampering
- Information Disclosure
- Denial of Service

The STRIDE Threat Modelling process identified a critical flaw in the applications architecture. This test verified the existence of that flaw which if exposed would leave the system vulnerable to tampering, information disclosure or denial of service.

7.2 Test: Unauthorised Direct Access to Server

Description

The APP X client software performed all user authentications and assessed user privileges within the application. It was identified during the case study Threat Model that a connection between the client and server was established using a client/server handshake and that no mutual authentication techniques were implemented.

With the right knowledge of client / server communication it is possible to bypass the client and connect directly to the server (Figure19). If an adversary were to do so they could attack the server directly or theoretically, perform more sophisticated attacks against the database through the server.

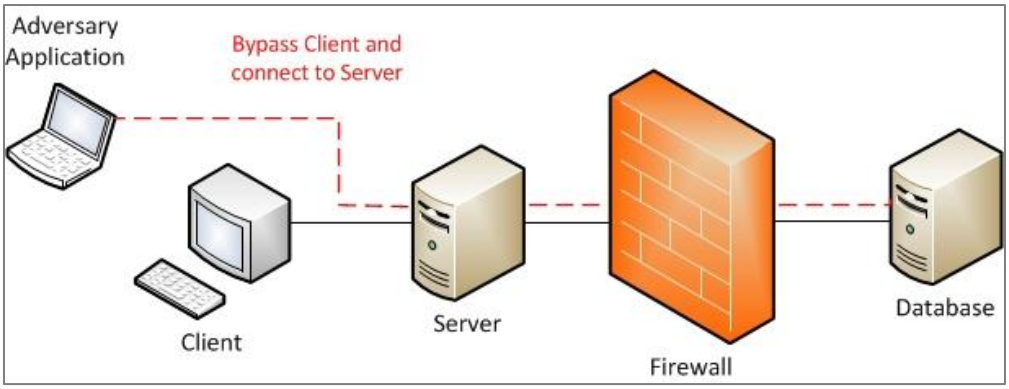


Figure 20 - Connection to Server Bypassing Client

It was the aim of this test to demonstrate how a potential attacker could connect to the applications server and perform a Denial of Service Attack.

Assumptions

- Adversary has access to the private network.
- Adversary must acquire Server IP.
- Adversary must acquire Server ports.
- Testing was carried out on the live production system.
- Detailed steps to acquire Server IP and ports are beyond the scope of this test.

Implementation

- An application was created to connect to the APP X Server mimicking a Client. (Appendix 6 – WinSock Script)
- The app was configured to take the credential entered (Figure 21) for IP address, Server and Event Ports and use the details to make a connection with APP X Server.

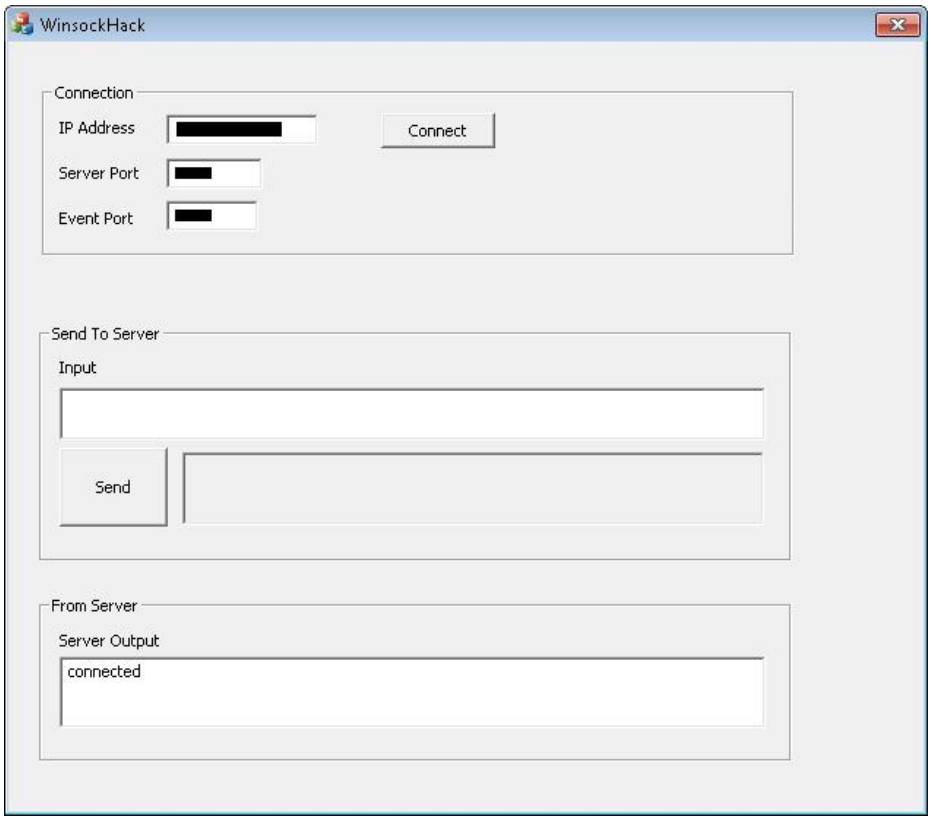


Figure 21 - Winsock Hack Application

- A socket was created using the C++ socket() system call (Listing 2). The socket was then connected to the server address using the connect() system call.

```
// Create a SOCKET for connecting to server
m_ConnectSocket = socket(result->ai_family, SOCK_STREAM, result-
>ai_protocol);

...

// Connect to server
iResult = connect( m_ConnectSocket, ptr->ai_addr, (int)ptr-
>ai_addrlen);
if (iResult == SOCKET_ERROR)
{
      closesocket(m_ConnectSocket);
      m_ConnectSocket = INVALID_SOCKET;
}
```

Listing 2 – Connection to APP X Server

- The Denial of Service was attempted by creating multiple threads simultaneously connecting to the server. In each thread a continuous loop then sent 10 bytes of data to the server every second.
- The test application was limited by the number of bytes that could be sent as a larger amount caused the test application to crash when run with a large number of threads.
- The test application was limited by the number of sockets that could be created before the test application crashed. The number of threads was set to 500.
- The application was run simultaneously 30 times on one machine resulting in 15,000 connections to the Server.
- APP X was running on the production system and monitored throughout.

Results

- The application successfully demonstrated how an adversary could connect to the APP X Server.
- The test resulted in affecting the performance of APP X. The response time of the application when performing operations was noted significantly slower.

The aim of this test was to demonstrate a method to connect to the APP X Server and perform a denial of service. The test successfully connected to the server and affected the performance of the application but did not cause the server to crash. The difficulty was the number of sockets that could be created and the number of bytes sent by each thread. The results suggest that performing a denial of service attack against the systems server is possible, but would require a distributed attack involving multiple machines on the network.

STRIDE Classification

Successful implementation of test scenario could result in:

- Denial of Service

The STRIDE Threat Modelling process identified that no mutual authentication existed between client and server. This test verified the existence of that security vulnerability. The focus of this

test was a denial of service attack. More sophisticated attacks could be performed against this vulnerability.

7.3 Test: Network Traffic Interception

Description / Definition

The Threat Model process identified that no data encryption is performed by APP X. Data packages sent can include information such as login credentials, SQL statements as well as IP addresses and port numbers. The vulnerability existed that unencrypted packets could be intercepted by an adversary (Figure 17). These packets may contain information that possibly could be used to gain access to the application or access at a higher level of privilege, perform man in the middle type attacks and/or help build a comprehensive picture of the database scheme by recording and analysing unencrypted SQL statements.

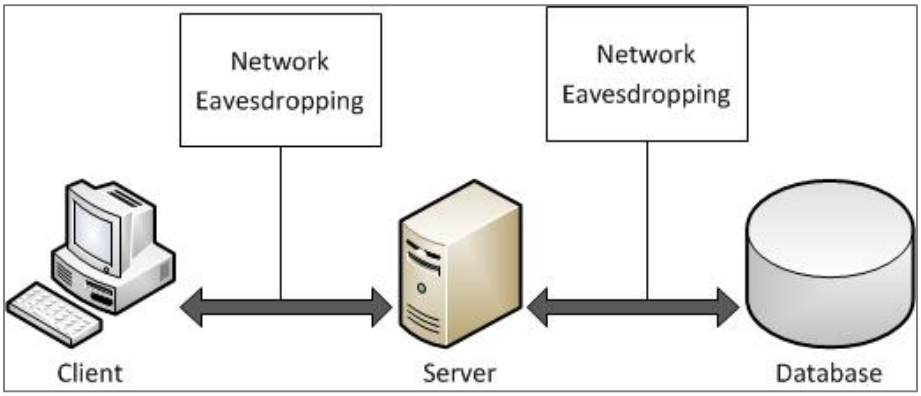


Figure 22 - Network Eavesdropping

It was the aim of this test to demonstrate one method of intercepting packages sent by APP X and to obtain data from those packets that could be used to perform more sophisticated attacks.

To perform this test Wireshark a popular Open Source network analyses program was used. This tool provides microscopic analysis of packets and protocols which can be captured and studied offline/offsite. Wireshark works by putting the NIC (Network) card in promiscuous mode to capture packets. It uses protocol dissectors to understand the various protocols and

has two libraries which help identify protocols or packets: WinPcap (Windows) and LibPcap (Linux) [43]. The tool provides highly configurable filter and capture options allowing packages on a particular port or from a particular IP Address to be captured.

Assumptions

- Adversary has access to the private network.
- IP addresses of application client, server and database have been captured. This is not a requirement. For the purpose of this test providing the correct IP's allowed the packets that were captured to be filtered for analyses.
- Testing was carried out on live production systems.
- Using Wireshark an adversary could capture packets and perform analyses at a later date and offsite.

Test Steps

- Wireshark was downloaded [44] and configured to run off a USB flash drive.
- The tool 'capture feature' to record network packets was started.
- APP X Client and Server were run.
- Some basic operations were performed on the application's Client.
- Wiresharks capture was stopped.
- The summary of the capture was viewed (Figure 19) identifying 1245 packets captured in 53 seconds.
- Wiresharks display filters helps reduce the number of packets to investigate. The server and database IP addresses were applied to the tools filter (Listing 33). This reduced the number of packets for analyses.

```
( Ip.addr == 10.60.100.152 ) && ( Ip.addr == 10.60.100.151
```

***IP Address alter for this report**

Listing 3 – Applied filter to wireshark packet list

- Each packet included information relating to: source and destination IP addresses, physical layer, datalink layer, transport layer and the data. These packets were analysed for unencrypted data sent from the applications server to the database.

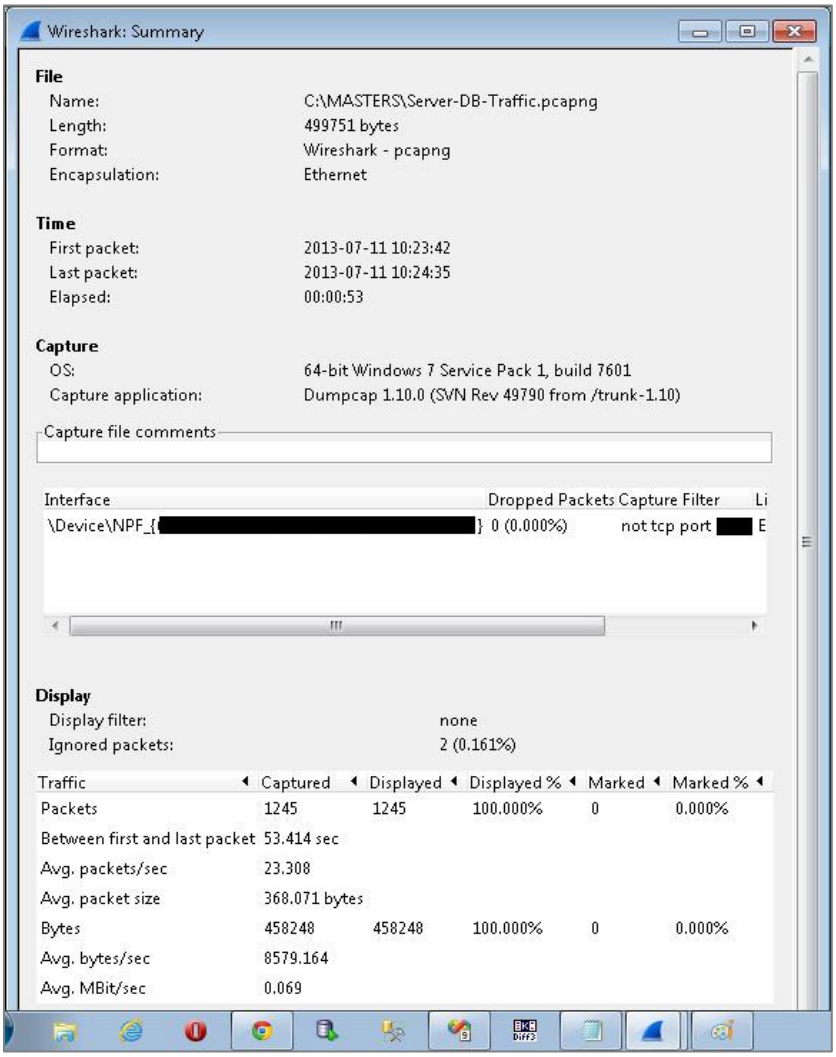


Figure 23 - Wireshark Capture Summary

Results of test run on live production system

- Wireshark successfully ran from a USB flash drive and captured packets sent across the live testing environments network.
- Wireshark successfully reduced the number of packets for analyses when the source and destination IP addresses were entered into the tools filter option.

- When assessing the filtered captured data a number of packets were identified to include SQL commands in plaintext (Figure 19).

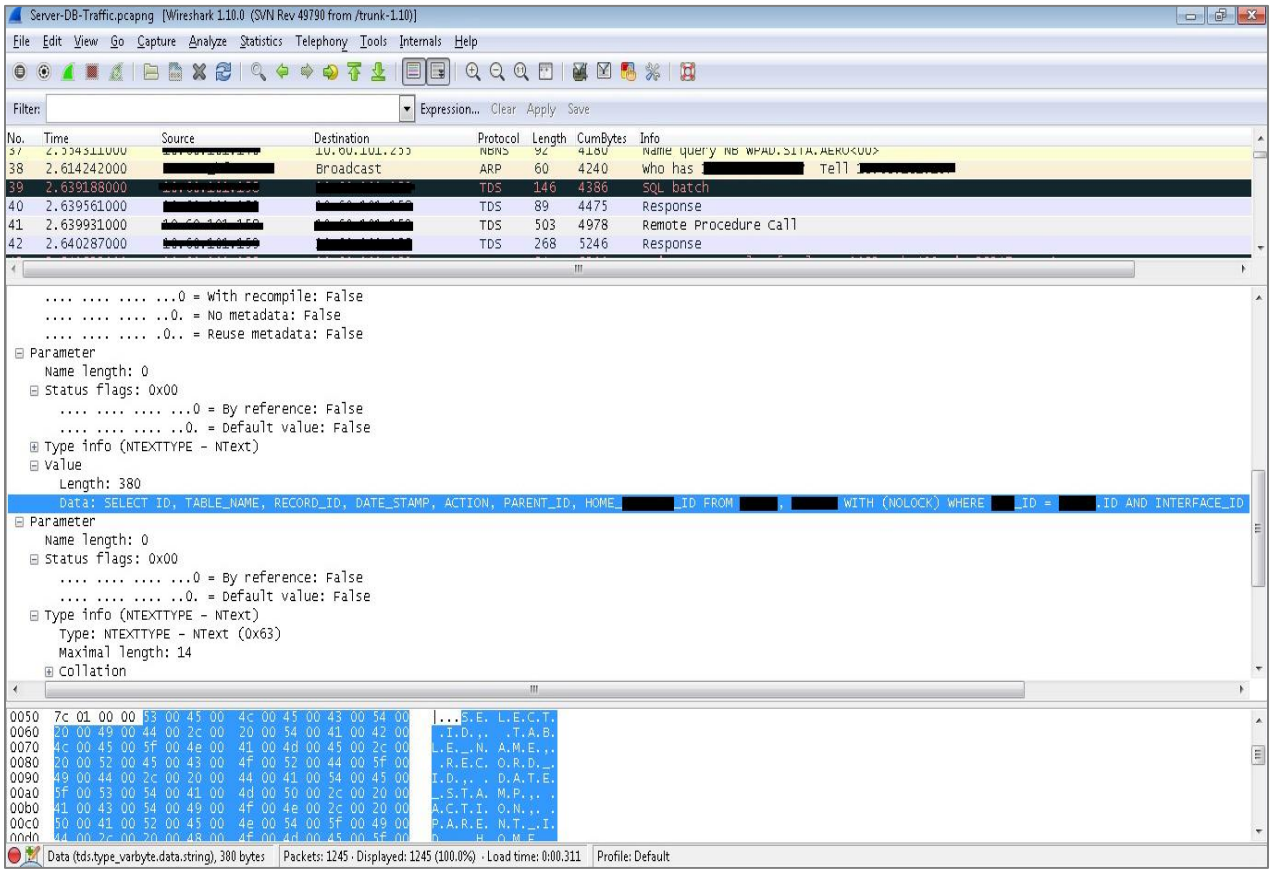


Figure 24 - Packet with SQL command in plaintext

The aim of this test was to demonstrate a method to capture unencrypted data sent across the network by APP X. These packets contained unencrypted SQL commands. This test was completed successfully. From here an adversary could begin gathering information about the database structure as well as acquire other user's credentials, IP address and Ports. This information could be used to perform more sophisticated or destructive attacks.

STRIDE Classification

Successful implementation of test scenario could result in:

- Tampering
- Information Disclosure

Working through the analyses of STRIDE per element, one of the first vulnerabilities identified in this case study was the sending of unencrypted data across the network. STRIDE provided a classification of tampering and Information disclosure for this vulnerability. Intercepting SQL statements could help an attacker build up a better understanding of the database. This validates the Information disclosure STRIDE classification. If an adversary intercepted packets of data it is possible in combination with test 7.2 that tampering could occur.

7.4 Additional Verification

Throughout the Threat Modelling process ad hoc testing was performed to verify assumptions made and issues identified. Such testing included:

Confirmation of Un-vetted External Interface / Access without Auditing

External interfaces can interact and amend data within the applications database. This was identified during the Threat Model process. Although these external applications are beyond the scope of this Threat Model, their interactions with the database are within scope. STRIDE identified the system as vulnerable in the event an interface is compromised. To validate that this vulnerability exists one external interface was tested. Changes to the database were performed. No logging exists that records external interface interactions. No records exist in the database that identifies which interface made changes to the database. This was validated by code and database checks. Introducing Interface logging would in the event of a hack or irregular behaviour, help the team narrow down / identify the source of the issue.

Sufficient auditing is currently not in place. If an adversary successfully compromised the system there would be little evidence to indicate where the attack came from or if an attempted hack was performed. This was tested and confirmed during analyses.

Input Injection

Injection type attack such as cross scripting or SQL injection were tested for during the Threat Model process. The applications validation library successfully sanitises all user input.

Buffer overflow

The application was tested for buffer overflow weaknesses. The applications validation library successfully sanitises all user input.

7.5 Conclusion

The testing outlined in this chapter confirmed that the main vulnerabilities identified by STRIDE could be exploited, thus validating the STRIDE Threat Models ability at identifying security risk. This chapter also summarised additional testing that was performed during the Threat Model process confirming successful mitigations were in place. Ethical hacking performed after the Threat Model process, provides developers the opportunity to implement techniques an adversary could use to attack the system. This knowledge can then be used to implement appropriate mitigations.

CHAPTER 8.

RESULTS / FINDINGS

Discussed in this section are the issues encountered when Threat Modelling a legacy enterprise application, the vulnerabilities identified by STRIDE / DREAD and the testing performed to demonstrate basic methods to exploit these vulnerabilities.

8.1 Threat Modelling of a Legacy Application

The case study conducted during this research produced a Threat Model of a large scale system using the STRIDE / DREAD methodologies. The system incorporated the main application deployed in a 3 tier approach (Client / Server / Database) with additional external interfaces including third party interfaces and additional interfaces developed by Company X. The size and heterogeneous nature of the systems encapsulated within the enterprise made it difficult to identify a starting point to the Threat Model. The Threat Model was scaled back to focus on the client, server and database components of the main application. Defining the scale of the Threat Model resulted in a more manageable project. Each external interface was identified at this point as requiring an individual Threat Model.

The legacy nature of the system resulted in limited documentation available. Producing dataflow diagrams of the system proved difficult. Andreas Schaad and Mike Borozdin stated in their research on Automated Threat Analysis that “even little additional information on architectural diagrams can yield significant value” [45]. The experience gained during this case study validates this argument. The success of a Threat Model is directly linked to the accuracy and detail of the systems DFD’s. To create a more accurate representation of the system the original Threat Model prerequisites were defined. This included documenting: system use scenarios, external dependencies, security notes, assets, entry/exit points and data flow diagrams. The gathering of this information was reliant on a number of sessions with the applications lead developer and his (extensive) knowledge of the system. If was found that his participation in the Threat Modelling process was necessary due to the lack of documentation. Some of these threat modelling sessions were also attended by a senior developer from the

applications support team. During these sessions (which included the constant reviewing and refining of the data gathered) the application was successfully documented and the high level DFD's of the system's architecture produced.

This research found that successfully Threat Modelling a legacy system is dependent on detailed information being available of that system. Legacy projects may not have detailed documentation available, but extensive in-depth information of that system may exist in the knowledge base of the people who currently support the project. This was the scenario encountered during this case study.

8.2 Results of STRIDE / DREAD

STRIDE was applied to each element of the systems dataflow diagrams, with the aid of Microsoft's Threat Modelling tool. Each classification of threat type against each element was investigated. Threats identified could be grouped into three types: Mitigated Threats, Unknown/unmitigated threats and Known/unmitigated threats.

It was identified during this analyses that the systems extensive validation libraries and system rules prevented users from performing operations beyond the intended use of the application. The main vulnerabilities identified were the result of the issues with the existing system architecture, the absence of data encryption of network traffic and inadequate auditing.

The implementation of STRIDE identified areas of the system without correct security mitigations. This information was analysed and from it a vulnerability list produced. This step proved to be the most subjective in the Threat Model process. The vulnerabilities identified included: potential unauthorised direct access to Database, unauthorised connections to the systems server, Un-vetted external interfaces and the possibility to intercept unencrypted network traffic. . It was felt during this Threat Model that the original developers of APP X underestimated insider threats. Dodge, Ferguson and Cappelli defined the threat from insiders, malicious or unknowingly, as "threats introduced to an organisation by a trusted entity" [46]. This type of threat poses the greatest risk level to APP X.

The DREAD risk analyses methodology was applied to the identified vulnerabilities. The process categorised each risk (Table 23, Chapter 6). The ratings produced by DREAD included one high risk, four medium and one low risk vulnerability. Scoring the risks with the use of DREAD produced similar results. The goal of DREAD was to produce a prioritised list of vulnerabilities to aid developers decide which risks to address first. The priority list was successfully compiled. However the results which were limited in range (one high, one low risk, with the remaining awarded a DREAD rating of medium) made it difficult to determine which risks to mitigate first, based solely on this data. This research carried out ethical hacking based on the findings of STRIDE in an attempt to confirm ease/complexity of exploiting the tested vulnerability.

The results of this Threat Model when compared to the 2013 top ten database threats (Table 24) as reported by Imperva suggest that APP X is vulnerable to at least four of those threats types including: Privilege abuse, Weak Audit Trail, Exploitation of Vulnerabilities/Misconfigured Databases and Denial of Service.

Ranking	2013 Top Threats
1	Excessive and Unused Privileges
2	Privilege Abuse
3	SQL Injection
4	Malware
5	Weak Audit Trail
6	Storage Media Exposure
7	Exploitation of Vulnerabilities and Misconfigured Database
8	Unmanaged Sensitive Data
9	Denial of Service
10	Limited Security Expertise and Education

Table 25 - Imperva 2013 Top Ten Database Threats [47]

These threat types were demonstrated by simulating potential attack techniques against the system.

8.3 Validation of Identified Vulnerabilities

The results of STRIDE / DREAD were tested, verified and presented to the APP X development team along with the applications Threat Model. This was achieved by implementing techniques to exploit the systems vulnerabilities. This step was beyond the scope of the Threat Model process. However, by demonstrating potential methods an adversary could use to attack the system, it was hoped to aid the applications support team in the implementation of appropriate mitigations. The demonstration of attack techniques combined with the DREAD ratings aided decision making. This is only possible for existing systems and not for Threat Models of applications in the design phase of the SDL.

The focus of these tests was to demonstrate how a potential adversary may attack the system. The ethical hacking also validated the vulnerabilities identified which in turn confirmed STRIDE's ability to identify security risks.

Testing proposed a method to gain access to the systems database by exploiting open ports on the networks firewall. Testing demonstrated a method to connect and communicate to the applications server by exploiting a lack of mutual authentication between the server and clients. Testing demonstrated the use of an open source tool (Wireshark) to intercept unencrypted data sent over the network. All test cases were successfully implemented on the live production environment.

Comparing the DREAD scores awarded to each vulnerability with the ease of implementing a technique to exploit those vulnerabilities suggests that DREAD scoring is relatively accurate. The 'Unauthorised Direct Access to Database' vulnerability was given a DREAD rating of High. The ethical hacking demonstrated the simple steps to exploit this vulnerability. By demonstrating the ease of performing such an attack and factor in the consequences if that attack was successful (direct access to the database), the exercise validates the HIGH DREAD rating.

8.4 Conclusion

The results of the STRIDE process successfully provided a systematic method to analyse APP X. The legacy nature of the system made initial steps difficult as little documentation was available. The initial steps were reliant on Threat Modelling sessions with members of the applications existing development team to acquire the necessary data. Once this information was successfully obtained, working through the STRIDE approach produced a list of vulnerabilities. Each decision made on potential vulnerabilities was documented with the use of the Threat Modelling tool. Some of the results of the STRIDE activity identified were issues the current development team were both unaware of, proving the benefits of Threat Modelling. Verification of a number of these vulnerabilities was performed by demonstrating how they could be exploited.

Threat Modelling identifies vulnerabilities with no current mitigations in place. It is the responsibility of the development team to use this information to put in place the appropriate countermeasures. Performing a Threat Model on a legacy system identified issues that should not exist in new projects. Vulnerabilities such as the direct database access identified in the case study may possibly require architectural changes. The cost to do so may be high. Whether a company allocates the necessary resources to make these changes or accepts the risk is an internal matter for that company. This issue does highlight the benefit of introducing Threat Modelling early in the development lifecycle. If such methodologies were available such vulnerabilities could have been avoided. Shoemaker and Ingalsbe [48] suggested that over 50% of defects occur during requirements engineering. Their theory is supported by this case study, in that the main defects identified were indeed architecture-related.

CHAPTER 9.

INDUSTRY FEEDBACK

Since implementing and presenting the case study Threat Model to company X, the practice of Threat Modelling has been introduced into the company's security practices. The following testimonial was provided Mr. N. the lead technical developer on this project.

"*Within our company the technical lead developers have been tasked to review and appraise the security of its systems from a software and hardware perspective. My remit involves providing a system architecture overview of APP X and a Threat Model that identifies weaknesses or areas of improvement. Michael's threat modelling project is pivotal to exposing potential security weakness and concerns to the development team, our customers and external vendors. We have reviewed the Threat Model and understand it to be comprehensive in nature, highlighting key points that need to be addressed and would see it as the de facto standard for all of the threat modelling required by our department. I would also like to share the Threat Model with our security advisor and see how it aligns with CMM.*"

Further feedback was acquired from other lead developers at Company X. In a discussion[5] with Mr G. (Lead developer on a different project), he explained that he was led to believe that "a threat model should take no longer than two hours". Feedback from the developers confirmed many common misconceptions, and highlights the need for more information, training, research and clarity of purpose within the industry. For the successful implementation of Threat Modelling, software developers must fully understand how the methodology operates.

[5] Michael McGrath in conversation with Mr. G, June 7th 2013

CHAPTER 10.

DISCUSSION

This research has investigated Threat Modelling as a security testing technique and confirms the efficiency of STRIDE as a comprehensive methodology for assessing software security vulnerabilities in legacy enterprise applications. The adoption of Threat Modelling by companies depends upon the individual preference of an organisation and their security requirements. Threat Modelling is liable to misconceptions, as outlined in chapter 9. Furthermore, Threat Modelling is also a relatively new practice. Few managers, developers and security personnel have these skills. The effectiveness of these processes relies upon an informed and knowledgeable application.

During initial research, multiple companies were approached about current security practices. Two companies, currently maintaining and developing legacy enterprise systems, have adopted different approaches to identifying software vulnerabilities. Company X, who provided access to their legacy system for this research case study, is now employing the offensive approach of Threat Modelling. Identifying and mitigating software vulnerabilities in both new and legacy projects. Company Y, another Irish based software company, is taking a defensive approach of penetration testing to secure their applications.

Network and application penetration testing requires a certain level of expertise. Both approaches may be time consuming. Threat Modelling can be a laborious exercise, while penetration testing requires an investment in initial training.

Ingalsbe, Kunimatsu and Beaten [49] stated that Threat Modelling is only one point on the broader risk management continuum. It is the opinion of this author that software security benefits from the adoption of multiple security disciplines. This research demonstrated that Threat Modelling and penetration testing can complement each other. Vulnerabilities were identified in the legacy system by implementing STRIDE. Penetration testing was then

performed, simulating attacks against the system. This type of testing can also be used to verify that mitigations have successfully been put in place.

As the case study in this research proceeded, there was a notable increase in the enthusiasm of Company X towards the potential of Threat Modelling. Mr N, the team's lead developer, initially supported the case study as an academic project. As the project proceeded through the various stages of analysis, there was a noticeable shift in Mr N's response. Mr. N recognised that through his support and interaction with this research, his own understanding and appreciation of the benefits of Threat Modelling became clear. What began as an academic exercise became a valuable contribution to Company X, and the benefits of Threat Modelling were circulated during in-house meetings.

It is important that all team members involved in Threat Modelling fully embrace the process. It is also essential that management incorporate it as part of the development lifecycle, and not just a once-off task. Threat Modelling can be a time consuming process, especially on legacy enterprise systems. The cost may deter some organisations from adopting such practices. Those companies need to determine the cost of implementing security processes such as Threat Modelling versus the impact of a successful attack against their system. Such breaches can result in financial losses and a loss of customer confidence. Company X, are proactively addressing the security issues addressed in this research and have introduced Threat Modelling as part of their SDLC. The security of their products will improve as a result.

The participation in this research by Company X suggests they are an organisation seeking to improve on current practices. It is, therefore, no surprise that Company X hold CMMI (Capability Maturity Model Integration) maturity level 2 certification. Company X adheres to CMMI for developers (CMMI-DEV) which provides guidance for applying best practices to improve the quality of the product/service developed [50]. Von Wangenheim, Hauch and Von Wangenheim stated [51] that in a competitive global market organisations involved in bidding contracts benefit from CMMI appraisal. The combination of CMMI maturity certification and the company's interest in the adoption of new practices such as Threat Modelling will benefit and strengthen their position in their specific market. Threat Modelling has been introduced by

Company X to manage, monitor and mitigate risks/threats. Threat Modelling provides Company X with the ability to successfully accomplish the risk management requirements for CMMI level 2.

Threat Modelling is not a requirement for CMMI or other standards such as ISO 9001 (standard for quality management systems). However, organisations can use Threat Modelling to improve their software security, one important aspect of CMMI and ISO certification.

CHAPTER 11.

SUMMARY

The last decade has seen an increased effort by the software industry to develop new practices and methodologies that include security in the development lifecycle of applications. Threat Modelling is one such practice that outlines a structured approach to identify and classify threats within a software system. By identifying and documenting threats, the appropriate countermeasures can be implemented in a more systematic way.

This research identified different Threat Model approaches and ascertained which methodology was most adapt for enterprise application development. This research then presented an in-depth case study of a Threat Model developed for an existing legacy enterprise system. Finally this research verified the results of STRIDE by demonstrating methods to exploit the vulnerabilities identified.

This research explored some of the current leading secure software development frameworks including Open SAMM, CLASP, BSIMM and SDL. These frameworks incorporate Threat Modelling as part of their development lifecycles and therefore merited close examination.

Threat Modelling has been established as a key component within leading secure software development frameworks. The SDL framework is a product of Microsoft's "Trustworthy Computing" initiative and is regarded as an industry leader in secure software development. The STRIDE Threat Modelling methodology is a product of the SDL framework. SDL is progressive, continuously refining its framework and Threat Modelling practice.

The fundamental concepts of Threat Modelling and the different available methodologies, including STRIDE, TRIKE, P.A.S.T.A and the AS/NZS 31000:2009 have all been investigated in this research. Each of the methodologies display similar characteristics. They attempt to formulate a reliable, repeatable approach to identify and rate security risks to software. The selection of an appropriate methodology was guided by the apparent strengths considered most relevant

for Threat Modelling. P.A.S.T.A and the AS/NZS 31000:2009 were dismissed as flawed methodologies. STRIDE and TRIKE were explored in detail.

As a methodology STRIDE approaches Threat Modelling from the viewpoint of an adversary and how an attack against a system may be performed. TRIKE approaches Threat Modelling from a defensive risk management perspective. This research revealed that no comparison between TRIKE and STRIDE has been published. Chapter 4 offers this comparison, and recognises STRIDE as a more comprehensive Threat Modelling methodology for assessing software security vulnerabilities in legacy enterprise applications.

A mock system depicting a basic web application was Threat Modelled by two variations of STRIDE and the TRIKE methodology v1.5. STRIDE was examined both manually and with the use of the Threat Modelling tool. The manual STRIDE approach followed the steps outlined in the book, *Threat Modelling* by Swiderski and Snyder. [52] This approach required an in-depth analysis of the system before implementing STRIDE. This exercise demonstrated STRIDE as independent of its tools. The Threat Modelling (TM) tool approach focused on the system's architecture with the aid of data flow diagrams and the automation of STRIDE-per-element analyses of these diagrams.

This research suggests the Threat Modelling tool could yield faster results, beneficial for enterprise systems. With relation to a legacy system, existing documentation may be limited. Working through the manual steps to depict and document the system before implementing STRIDE may be beneficial to such projects

This research presented a case study Threat Model of a legacy enterprise application, implemented with the use of STRIDE and the supporting Threat Modelling tool. A risk analysis was performed by DREAD. The main challenges faced during this case study were related to the lack of detailed documentation of the application, a common complaint of legacy systems.

This research concluded with verification of the vulnerabilities identified by STRIDE. The aims of these tests were to identify potential methods an adversary may use to exploit the security

vulnerabilities presented by the Threat Model. These tests confirmed the findings of the STRIDE process to be accurate. Each risk tested was successfully exploited.

CHAPTER 12.

CONCLUSIONS & FUTURE WORK

The findings of this research work lead the author to conclude that STRIDE is a more comprehensive Threat Modelling methodology for assessing software security vulnerabilities in legacy enterprise applications.

The following conclusions (in no particular order) were arrived at.

12.1 Comprehensive tools

Threat Modelling can take time to complete. To assist, supporting tools have been developed. Enterprises must carefully consider the methodology and the tools available to support that methodology. Microsoft's 'Threat Modelling Tool' is an example of a comprehensive, cost efficient and easy to use tool to support the STRIDE Threat Modelling process. This tool provides DFD validation and auto-generates threats.

This research presented a comparison of STRIDE V TRIKE Threat models. When performing this comparison both the Threat Modelling Tool and the TRIKE spreadsheet were used. STRIDE's tool identified threats more quickly and increased the speed in which the Threat Model could be performed. It also displayed data in a more user friendly manner.

TRIKE is dependent on the TRIKE spreadsheet to implement its methodology. However, the TRIKE spreadsheet is not fully developed with placeholders for future features and lacks the tutorials and documentation available for the STRIDE Threat Modelling tool. The TRIKE spreadsheet also demonstrates an over-reliance on colour to represent different data. Visually impaired people may struggle to distinguish between the TRIKE colour schemes, as was confirmed by a colour-blind colleague who struggled to use the spreadsheet in its intended way.

The Threat Modelling tool further demonstrated its efficiency during the Threat Model case study of a legacy enterprise application, presented in this research. The large volume of data

was maintained and presented with the aid of a clear user interface. This research has verified the benefit this tool offers in assisting STRIDE Threat Models enterprise applications.

12.2 Legacy Systems

The maintenance and enhancement of Historical/Legacy systems provide their own specific challenges. They remain in use for a variety of reasons including complexity and the cost of redevelopment, or simply because the product owner has seen no reason to replace the application. Information on the unique challenges of Threat Modelling these systems was scant.

The main challenges faced during the case study presented in this research were related to the lack of detailed documentation of the application, a common flaw in legacy systems. The focus of the current STRIDE approach is to diagram, identify threats, mitigate and validate. This is adequate for new projects or systems where requirements and design documentation are readily available to assist produce accurate dataflow diagrams. Sparse documentation makes the creation of sufficient DFDs a difficult task. To overcome this challenge with a large legacy system the original Threat Modelling prerequisites such as defining use scenarios and system entry points, were collated, so the system could properly be understood. This information was retrieved from the knowledge base of the current development team.

This research found that successfully Threat Modelling a legacy system is dependent on the availability of detailed information of that system. Legacy projects may not have sufficient documentation available, but extensive in-depth information of that system may exist in the knowledge base of the people who currently support the project. The original STRIDE steps are an excellent starting point to extract that information from the current team and use that information to build up a greater understanding of the subject system being Threat Modelled.

12.3 Subjective Assessment – STRIDE / DREAD

Correctly knowing when to stop Threat Modelling (exit strategy) is one subjective aspect of STRIDE. For example when modelling the system different levels of DFDs were created starting with a high level context diagram and from there producing a hierarchy of detailed DFDs.

During analysis, Injection type attacks were identified as sufficiently mitigated by the applications validation libraries. This was documented against the higher level DFDs. At the lower level DFDs it was felt the analyses became repetitive. Threats generated by STRIDE were mitigated by the same techniques previously documented at a higher level. By identifying repetition at this point the modelling was stopped. This subjective aspect of STRIDE had little impact on the methodologies ability to identify vulnerabilities. The sequential steps, well-defined threat classifications, extremely competent support tools allow STRIDE to repeatedly identify security vulnerabilities. Knowing when to stop modelling and start analysing may come with experience.

The DREAD methodology was used to analyse the risks identified by STRIDE. Risk assessment is an important final step in Threat Modelling. DREAD provided an easy to implement technique to rate each risk. This research discussed two variations of the DREAD marking system. The original marking scheme for each step in DREAD was based on values between one and ten for each category of DREAD. This scoring system was subjective with varying results. For example security personnel may award a higher mark than developers who may not deem the same vulnerability as high a risk. The end list of prioritised vulnerabilities was heavily influenced by the individual(s) awarding the DREAD score. The refined version of DREAD provided a marking scheme of values between one (lowest) and three (highest) which were awarded based on the severity/impact of that vulnerability in each of the dimensions of DREAD. This version of DREAD was used during the case study presented. The resulting prioritised list awarded the majority of vulnerabilities with a medium risk rating following the new marking scheme. Such little variation in rankings resulted in a prioritised list which didn't really prioritise the vulnerabilities identified by DREAD.

12.4 Enterprise Applications

This case study in this research presented a Threat Model of an existing enterprise system. Due to the size of the system it was initially difficult to identify a starting point. STRIDE provided a step to clearly define the scope of the Threat Model, focusing the case study on the main components of the system.

A security issue with enterprise systems is the existence of external interfaces and varying third party applications that can interact with the system. STRIDE can incorporate external applications without generating threats. This is beneficial as assumptions and security notes can be documented and appropriate security measures put in place on the systems entry point.

The size and nature of enterprise applications can result in variations of complexity and quality of code. Numerous developers may work on a system. These developers may have varying degrees of competencies and security techniques may not be adequately implemented. The STRIDE methodology is code independent. The process systematically evaluates a software system and identifies unmitigated vulnerabilities. This repeatable practice greatly improves enterprise application security.

12.5 Industry Perception

Threat Modelling requires additional resource allocation during project planning and development. As a result the process may encounter initial resistance from developers who remain functionality oriented. Some Threat Modelling critics dismiss the process as too time and labour intensive. These criticisms originated in early Threat Modelling efforts. Refinement of methodologies and the advancement of supporting tools now place Threat Modelling as an appropriate practice for identifying and managing risks/threats. As companies aim for CMMI and ISO certifications, new practices and processes are required. Reluctant developers may have no choice but to adopt practices like Threat Modelling. It is the opinion of this author that only through practical implementation can software professionals acquire a better understanding and appreciation of Threat Modelling and the benefits the process can bring to application development.

12.6 Security Vulnerabilities

Threat Modelling provides software developers a reliable, repeatable approach to identify and rate security vulnerabilities in software. However, this security practice tends to focus on new projects and incorporating Threat Modelling early into the development lifecycle. The case

study presented in the research highlighted the benefit of apply the STRIDE Threat Model methodology, by identifying critical vulnerabilities within the system.

There is a misconception that if a system is large, then it is secure. Quite often, legacy enterprise systems are more critical as the system has not been modelled. Forgotten vulnerabilities may exist. STRIDE can identify unknown threats or vulnerabilities that have potentially been ignored.

In conclusion, this research has presented Threat Modelling as an efficient practice in identifying potential vulnerabilities within software systems. The case study presented pitched the STRIDE methodology against a real world legacy enterprise application. The historical nature of the project provided its own unique challenges, however STRIDE proved to be a comprehensive methodology to Threat Model a legacy enterprise application. The process provided well-defined threat classifications, extremely competent support tools and recommendations of mitigation techniques based on threat type. The volume of resources and the ease to produce good quality Threat Models are the reason this methodology is a current[6] industry leader.

12.7 Future Work

This research established that there are areas of the Threat Modelling process that warrant further investigation. Threat assessment requires further research and development. This process is not a simple task, as different factors will impact on the threat. DREAD was applied to the case study in this research. The results of this process were not comprehensive and did not provide a priority listing that would be of significant benefit to a development team.

Finally, during verification of the case studies STRIDE results, ethical hacking was performed on the application. This provided the opportunity to prove the results of the Threat Model while also demonstrating techniques of how a potential adversary might exploit the vulnerabilities

[6] June 2013

identified by STRIDE. Threat Modelling and penetration testing are very broad topics. However, further research into how the two interlink is a subject worthy of further analysis.

REFERENCE

[1] Gates, B. (2002). *Gates memo: 'We can and must do better'.* Available: httP://nehttp://news.cnet.com/2009-1001-817210.htmlws.cnet.com/2009-1001-817210.html. Last accessed 11th Dec 2012.

[2] Geer, D. (June 2010). Are Companies Actually Using Secure Development Life Cycles? *Innovative Technologies for Computer Professionals - IEEE Computing Society*. 43 (6), p12,16.

[3] Microsoft. (2013). *Security Development Lifecycle.* Available: http://http://www.microsoft.com/security/sdl/default.aspx. Last accessed 2nd June 2013.

[4] OWASP. (2010). *Software Assurance Maturity Model.* Available: http://www.opensamm.org/. Last accessed 10th Nov 2012.

[5] McGraw, G. Migues S. West J. (2012). *BSIMM4.* Available: https://buildsecurityin.us-cert.gov/bsi/resources/sites/1100-BSI.html. Last accessed 09th Nov 2012.

[6a] McGraw, G. Migues, S. West, J. (2012). *BSIMM Community.* Available: shttp://bsimm.com/community/. Last accessed 13th Feb 2013.

[6b] McGraw, G. Migues, S. West, J. (2012). *BSIMM Online.* Available: http://bsimm.com/online/. Last accessed 12th Feb 2013.

[7] Teodoro, N. Serrao, C. (June 2011). Web application security: Improving critical web-based applications quality through in-depth security analysis. *Information Society (i-Society), 2011 International Conference*. P457-462.

[8] OWASP. (2012) *CLASP (Comprehensive, Lightweight Application Security Process).* Available: https://buildsecurityin.us-cert.gov/bsi/resources/sites/132-BSI.html. Last accessed 08th Nov 2012.

[9] Gregoire, J. ; K.U. Leuven, Leuven ; Buyens, K. ; De Win, B. ; Scandariato, R. (2007). On the Secure Software Development Process: CLASP and SDL Compared. *IEEE - Third International Workshop on SESS,* Minneapolis, 2007.

[10] Microsoft. (2012). *What is the Security Development Lifecycle?* Available: http://www.microsoft.com/security/sdl/default.aspx. Last accessed 15 Oct 2012.

[11] Hussain, S. Erwin, H. Dunne, P. "Threat modeling using formal methods: A new approach to develop secure web applications," Emerging Technologies (ICET), 2011 7th International Conference on, pp.1,5, 5-6 Sept. 2011, doi: 10.1109/ICET.2011.6048492

[12] Microsoft. (2012). *SDL Threat Modeling Tool.* Available: http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx. Last accessed 28th Nov 2012.

[13] Kaspersky Lab. (2012). *IT threat evolution in Q3 2012: Microsoft no longer features among the top 10 products with vulnerabilities.* Available: http://www.kaspersky.com/about/news/virus/2012/IT_threat_evolution_in_Q3_2012_Microsoft_no_longer_features_among_the_top_10_products_with_vulnerabilities. Last accessed 10th Feb 2013.

[14] Agile Methodology. (2008). *What Is Agile?* Available: http://agilemethodology.org/. Last accessed 20th Feb 2013.

[15] Sullivan, B. (2008). *Streamline Security Practices For Agile Development.* Available: http://msdn.microsoft.com/en-us/magazine/dd153756.aspx. Last accessed 10th Oct 2012.

[16] Microsoft. (2012). *Melding the Agile and SDL Worlds.* Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ee790618.aspx. Last accessed 30th October 2012.

[17] Fagan, M. (2010). *POLL - What is your experience with security in the Software Development LifeCycle?* Available: http://erratasec.blogspot.ie/2010/02/poll-what-is-your-experience-with.html. Last accessed 08th Nov 2012.

[18] Jackson Higgins, K. (2010). *Survey Says: More Than Half of Software Companies Deploying Secure Coding Methods.* Available: http://www.darkreading.com/vulnerability-management/167901026/security/application-security/224200945/index.html. Last accessed 07th Nov 2012.

[19] Microsoft. (2012). *Introduction to the Microsoft Security Development Lifecycle.* Available: http://www.microsoft.com/en-us/download/details.aspx?id=16420. Last accessed 01st Oct 2012.

[20] Mockel, C. Abdallah, A.E., "Threat modeling approaches and tools for securing architectural designs of an e-banking application," Information Assurance and Security (IAS), 2010 Sixth International Conference. p149,154, 23-25 Aug. 2010 doi: 10.1109/ISIAS.2010.5604049

[21] Steven, J. "Threat Modeling - Perhaps It's Time," Security & Privacy, IEEE , vol.8, no.3, pp.83,86, May-June 2010 doi: 10.1109/MSP.2010.110

[22] Ockwell-Jenner Dave. (2012). A Structured approach to identifying, understanding and classifying threats to software. *Threat Modeling*. 1 (1), p01-42.

[23] Microsoft. (2012). *SDL Threat Modeling Tool 3.1.8.* Available: http://www.microsoft.com/en-us/download/details.aspx?id=2955. Last accessed 01st February 2013

[24] Hernan, S. Lambert, S. Ostwald, T. Shostack, A. (2006). *Uncover Security Design Flaws Using The STRIDE Approach.* Available: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx. Last accessed 22nd March 2013.

[25] Jeffries, C. (2012). Security Architect, Microsoft Services. *Threat Modeling and Agile Development Practices.* Available: http://technet.microsoft.com/en-us/security/hh855044.aspx. Last accessed 24th March 2013.

[26] Saitta, P, Larcom, B. Eddington M. . (July 13th 2005). *Trike v.1 Methodology Document.* Available: http://octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf. Last accessed 22nd March 2013.

[27a] Larcom, B. Saitta, E. (2008). *Trike.* Available: http://www.octotrike.org/home.shtml. Last accessed 05th Oct 2012.

[27b] Larcom, B. (2011). *HAZOP Analysis.* Available: http://www.octotrike.org/talks/baythreat2010.shtml. Last accessed 13th April 2013

[28] UcedaVelez, T. (2012). *Real World Threat Modeling Using the PASTA Methodology.* Available: https://www.owasp.org/images/a/aa/AppSecEU2012_PASTA.pdf. Last accessed 08th Oct 2012.

[29] Australian/New Zealand Standard. (2009). AS/NZS ISO 31000:2009 Risk management— Principles and guidelines. 3 (1), p01- 05.

[30] PTA. (2005). *Practical Threat Analysis for Information Security Experts.*Available: http://www.ptatechnologies.com/. Last accessed 01st February 2013.

[31] Security TechCenter. (2012). *Security Bulletin Severity Rating System.*Available: http://technet.microsoft.com/en-us/security/gg309177.aspx. Last accessed 09th April 2013.

[32] NIST. (2007). *Common Vulnerability Scoring System.* Available: http://nvd.nist.gov/cvss.cfm. Last accessed 19th March 2013.

[33] Microsoft. (2012). *Visio Overview.* Available: http://visio.microsoft.com/en-us/preview/default.aspx. Last accessed 16th March 2013.

[34a] Microsoft. (2012). *Security Development Lifecycle.* Available: http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx. Last accessed 22nd March 2013.

[34b] Microsoft. (2012). *Elevation of Privilege (EoP) Card Game.* Available: http://www.microsoft.com/security/sdl/adopt/eop.aspx. Last accessed 30th October 2012.

[35] Trike. (2008) *Tools.* Available: http://www.octotrike.org/tools.shtml. Last accessed 29th March 2013.

[36] Microsoft. (2011). SDL Threat Modeling Tool . *Manual*. Version 3.1.8 (1), p01.

[37] Trike. (2011). *Trike Help.* Available: http://sourceforge.net/apps/trac/trike/browser/spreadsheet/trunk/docs/help/TrikeHelp.xlsx. Last accessed 29th March 2013.

[38] Seidler, K. (2013). *XAMPP.* Available: http://www.apachefriends.org/en/xampp.html. Last accessed 07th July 2013.

[39] XAMPP (2013). *Download XAMPP LITE.* Available: http://sourceforge.net/projects/xampp/?source=dlp. Last accessed 01st July 2013.

[40] Microsoft (2013). *The SQL Server Driver for PHP....* Available: http://msdn.microsoft.com/en-us/sqlserver/ff657782.aspx. Last accessed 12th June 2013.

[41] Hiren, D. (2012). *Connect with Microsoft Sql server from PHP in xampp.* Available: http://davehiren.blogspot.ie/2012/01/connect-with-microsoft-sql-server-from.html. Last accessed 12th June 2013.

[42] Foundstone. (2013). *McAfee Free Tools.* Available: http://www.mcafee.com/us/downloads/free-tools/index.aspx. Last accessed 01st July 2013.

[43] Combs, G. (2013). *Whats on your network.* Available: http://www.wireshark.org/about.html. Last accessed 16th July 2013.

[44] Wireshark. (2013). *Downloads.* Available: http://www.wireshark.org/download.html. Last accessed 16th July 2013.

[45] Schaad, A. Borozdin Mike. (2012). TAM2: Automated Threat Analysis.*SAC '12 Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 1, p1103-1108.

[46] Dodge, R.C. Ferguson, A.J. Cappelli, D.M.. (Jan. 2013). Introduction to Insider Threat Modeling, Detection, and Mitigation Track. *2013 46th Hawaii International Conference on System Sciences*. 46 (no.1), p1812. d.o.i 10.1109/HICSS.2013.308

[47] Imperva. (2013). *Top Ten Database Threats.* Available: http://www.imperva.com/docs/WP_TopTen_Database_Threats.pdf. Last accessed 12th July 2013.

[48] Mead, N.R.; Shoemaker, D.; Ingalsbe, J., "Ensuring Cost Efficient and Secure Software through Student Case Studies in Risk and Requirements Prioritization," System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference, vol., no., pp.1,9, 5-8 Jan. 2009. doi: 10.1109/HICSS.2009.193

[49] Ingalsbe, J.A.; Kunimatsu, L.; Baeten, T.; Mead, N.R., "Threat Modeling: Diving into the Deep End," Software, IEEE , vol.25, no.1, pp.28,34, Jan.-Feb. 2008. doi: 10.1109/MS.2008.25

[50] CMMI Product Team. (2010). *CMMI® for Development, Version 1.3* .Available: http://www.sei.cmu.edu/reports/10tr033.pdf. Last accessed 22nd July 2013.

[51] von Wangenheim, C.G. Hauck, J. von Wangenheim, A.. (March-April 2009). Enhancing Open Source Software in Alignment with CMMI-DEV.*Software, IEEE*. 26 (2), p59 - 67. doi: 10.1109/MS.2009.34

[52] Swiderski, F. Snyder, W. (2004). *Threat Modeling*. Washington USA: Microsoft Press.

APPENDICIES

Appendix 1 – STRIDE Mitigation Sheet

THREAT	THREAT TYPE	DFD ELEMENTS
Spoofing	Authentication	**Authenticate principals (people or components):** Basic authentication Digest authentication Cookie authentication Windows authentication (NTLM) Kerberos authentication PKI systems such as SSL/TLS and certificates IPSec Digitally signed packets **Authenticate code or data:** Digital signatures Message authentication codes Hashes
Tampering	Integrity	Access Control Lists (ACLs) Digital signatures Message authentication codes
Repudiation	Non-repudiation services	Strong authentication Secure auditing and logging Digital signatures Secure time-stamps Trusted third parties
Information Disclosure	Confidentiality	Access Control Lists (ACLs) Encryption
Denial of Service	Availability	Access Control Lists (ACLs) Filtering Quota Authorization
Elevation of Privilege	Authorization	Access Control Lists (ACLs) Group or role membership Privilege ownership Permissions Privilege ownership Permissions

Appendix 2 – STRIDE Threat Model

Please refer to research files on disk for STRIDE Threat Model presented in Section 4.1

Appendix 3 –TM Tool

Please refer to research files on disk for Threat Modelling Tool files presented in Section 4.2

Appendix 4 – TRIKE Spreadsheet

Please refer to research files on disk for TRIKE Spreadsheet presented in Section 4.4

Appendix 5 – DB script

Please refer to research files on disk for further demonstration of PHP files used during testing. Section 7.1

Appendix 6 – WinSock

Please refer to the research files on disk for further demonstration on the WinSock code used to connect to APP X Server during testing. Section 7.2

Appendix 7 – An Introduction to Threat Modelling, Slides

An Introduction to Threat Modelling.

During this research a Guest lecture appearance, providing an introduction to Threat Modelling was presented to a group of LYIT Computer Science Students, March 2013

Two further publications are currently being written for submission to a leading journal.