2-SQUARE: WEB-BASED ENHANCEMENT OF PRIVACY AND SECURITY FOR SQUARE

Alan Ning-Fat Lai
B.S., University of California, Davis, 2006

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

SOFTWARE ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2011
2-SQUARE: WEB-BASED ENHANCEMENT OF PRIVACY AND SECURITY FOR SQUARE

A Project

by

Alan Ning-Fat Lai

Approved by:

__________________________, Committee Chair
Dr. Cui Zhang

__________________________, Second Reader
Dr. Senad Busovaca

__________________________
Date
Student:  Alan Ning-Fat Lai

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator   ___________________
Dr. Nikrouz Faroughi                Date

Department of Computer Science
Abstract

of

2-SQUARE: WEB-BASED ENHANCEMENT OF PRIVACY AND SECURITY FOR SQUARE

by

Alan Ning-Fat Lai

Software security and privacy requirements engineering can be a daunting task even with the proper expertise. The Security Quality Requirements Engineering (SQUARE) methodology was developed by the Software Engineering Institute at Carnegie Mellon University to aid with security requirements engineering. SQUARE was then later adapted for privacy requirements engineering. It is a nine step process that guides project teams to analyze and review security and privacy concerns using a structured methodology.

2-SQUARE is a web based tool for semi-automation of the SQUARE process for both Security and Privacy requirements engineering. This application handles many of the menial tasks such as organizing and maintaining an up to date set of artifacts developed throughout the process. Users are given interfaces designed for the tasks while also including guidance on the task at hand. As users enter artifacts and upload documents, all participants within a project are able to view up to date documents. 2-SQUARE also includes features to foster communication between project team members by allowing the easy sharing of comments and documents throughout the process. Finally 2-SQUARE was built to be easily expandable, it has the capabilities to easily
incorporate additional variations of SQUARE as well as individual modules for certain steps of the process.

The system is implemented in C# using the Asp.Net MVC framework with a Microsoft SQL database. The system will be deployed to a cloud based hosting service.

_______________________, Committee Chair
Dr. Cui Zhang

_______________________
Date
ACKNOWLEDGMENTS

The completion of this project would not have been possible without the support and guidance that I received. I would like to express my deepest gratitude to the following people:

Dr. Cui Zhang, for providing me the guidance I needed, not only with my project but throughout my time in the graduate program. Her knowledge, experience, and passion for teaching and software engineering have motivated me to share that enthusiasm with my fellow colleagues and use that knowledge to produce quality software.

Dr. Senad Busovaca, for taking the time to review my paper and for his valuable input.

Jennifer Lai, my loving wife, for standing by me and encouraging me to give my best to complete this project. I could not have completed this project or my degree without your support.

Bill and Ivy Lai, my parents, for raising me to be who I am today and encouraging me to continue with my education. Your continual love and support made the completion of this project possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>vii</td>
<td></td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2. BACKGROUND AND RELATED WORK</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2.1. SQUARE Methodology</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2.2. Privacy SQUARE Methodology</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2.3. mySQUARE</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2.4. CyLab SQUARE Tool</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2.5. eSQUARE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2.6. 2-SQUARE</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3. 2-SQUARE DESIGN</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3.1. Functional and Non-Functional Requirements</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3.2. 2-Square Architecture Design</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3.3. 2-Square User Interface Design</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>4. 2-SQUARE IMPLEMENTATION</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>4.1. View Implementation</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>4.2. Controller Implementation</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>4.3. Services Implementation</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>4.4. Data Model and Database Implementation</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>5. CASE STUDY</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>
6. CONCLUSION .......................................................................................................................... 36

Appendix. 2-SQUARE Requirements Document .................................................................. 37

References ................................................................................................................................. 47
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1. 2-SQUARE Software Architecture</td>
<td>11</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2. 2-SQUARE Workflow</td>
<td>12</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3. 2-SQUARE Navigational Bar, Categorization Step</td>
<td>14</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4. Authenticated User Home Page</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 5. Project Home Page</td>
<td>15</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 6. Notes and File Sharing Functions</td>
<td>16</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 7. List of Selected Definitions Page</td>
<td>17</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 8. Business and Security Goals Page</td>
<td>17</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 9. Privacy Asset and Goals Page</td>
<td>18</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 10. List of Artifacts Page</td>
<td>18</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 11. Security Requirements List Page</td>
<td>19</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 12. Requirements Categorization Page</td>
<td>20</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 13. Prioritization of Requirements Page</td>
<td>21</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 14. Inspection and Defect Tracking of Requirements</td>
<td>21</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 15. Controller Classes for SQUARE Variations Security and Privacy</td>
<td>24</td>
</tr>
<tr>
<td>16.</td>
<td>Figure 16. Add-On Modules (top) and Project Artifact (bottom) Controllers</td>
<td>25</td>
</tr>
<tr>
<td>17.</td>
<td>Figure 17. Database Tables for Process</td>
<td>27</td>
</tr>
<tr>
<td>18.</td>
<td>Figure 18. AMS Selected Definitions</td>
<td>30</td>
</tr>
<tr>
<td>19.</td>
<td>Figure 19. AMS Diagram Sharing</td>
<td>31</td>
</tr>
<tr>
<td>20.</td>
<td>Figure 20. Entering First AMS Requirement</td>
<td>32</td>
</tr>
</tbody>
</table>
21. Figure 21. Prioritization AMS Requirements ................................................................. 33
22. Figure 22. Requirements Inspection for AMS ............................................................... 34
23. Figure 23. Risks in Privacy Risk Analysis for Ubiquitous Computing Module .......... 35
Chapter 1

INTRODUCTION

The latest web-based applications encourage the sharing of information through social networks and ad networks, allowing companies to provide free services. These free services provide the convenience to access an individual’s own personal data from anywhere. However, because these applications are available over the internet, they have a significantly larger exposure to attacks and should be able to defend the user’s data from security and privacy breaches. Because of the sensitivity of information stored within databases it is essential to ensure that information is not shared without the approval of the owner.

Security Quality Requirements Engineering (SQUARE) is a methodology developed by the Software Engineering Institute’s Network Systems Survivability (NSS) Program at Carnegie Mellon University. It is designed to foster communication between software engineers and stakeholders to effectively elicit security requirements [1]. The SQUARE process has been further adapted for privacy requirements engineering with a proven structured method of producing a set of verifiable requirements [2]. While a general purpose requirements engineering process can be used, it can be beneficial to have a process designed to elicit privacy requirements.

This work presents 2-SQUARE, an integration and automation tool aimed at security and privacy requirements engineering using the SQUARE methodology. The 2-SQUARE system was developed utilizing the Microsoft .Net MVC Framework and Entity Framework. It is a web-based tool that allows multi-location collaboration and provides guided help for each role involved with the process. The tool helps requirements engineering teams work on security and privacy requirements engineering utilizing the SQUARE methodology from one application.
Chapter 2 of this paper discusses the background and related work leading to this project. Chapter 3 describes the design of the 2-SQUARE system. Chapter 4 covers the implementation of each of the layers for the application. Chapter 5 shows a case study demonstrating the improvements afforded by 2-SQUARE over existing methods. And finally Chapter 6 gives a brief summary and potential future work to enhance the abilities of 2-SQUARE.
Chapter 2

BACKGROUND AND RELATED WORK

2.1. SQUARE Methodology

The SQUARE process was designed as a method to elicit, categorize and prioritize security requirements. Its goal was to integrate security into the early stages of software development, but it has also proved useful for evaluating security of existing systems [1]. SQUARE guides teams through a series of nine steps in order to determine the project’s security requirements in a structured manner. Each of the following nine steps help project teams determine the necessary security goals and requirements:

1. **Agree on Definitions** – The first task is for stakeholders and the requirements engineering team to agree on a set of security related terms and definitions. This step reconciles the different perspectives of each member and ensures proper communication and understanding of all project and security related terms.

2. **Identify Security Goals** – The next step requires the stakeholders to first determine an overall business goal for the application and then compile and prioritize security goals. Different stakeholders may have different needs, and in turn, different security concerns. Understanding and prioritizing these security goals will allow the team to determine which security goals support the overall business goal.

3. **Develop Artifacts** – After understanding what the priorities are for the stakeholders, understanding the system and its potential vulnerabilities is necessary. In this step, artifacts such as system architecture diagrams, use case scenario/diagrams, misuse
scenario/diagrams, attack trees, standard templates and forms are collected from the stakeholders.

4. **Perform Risk Assessment** – Next, the project team must analyze the designs and information collected to determine potential risks. Any number of methodologies can be utilized depending on which one suits the situation best. The risk assessment helps the team find vulnerabilities and the likelihood of a particular threat. Understanding the potential threats will allow the team to determine proper protections later on in the process.

5. **Select Elicitation Technique** – In this step, the requirements engineers must select a requirements elicitation technique. Different techniques suit different situations–various factors can affect the usefulness of a specific technique such as the number of stakeholders, the expertise of the requirements team, and the size of project. Understanding the benefits and the drawbacks of various methods will help find the proper technique.

6. **Elicit Security Requirements** – After an elicitation technique has been selected, the team must execute the selected method. This step involves using a separate elicitation process from SQUARE, but it can take advantage of information that has already been collected. The final results from the chosen technique should produce a list of security requirements.

7. **Categorize Requirements** – Once all requirements have been collected each one must be categorized. At a minimum, it is suggested that the following categories should be used: Essential, Non-Essential, System Level, Software Level, and Architectural Constraint [1].
8. **Prioritize Requirements** – In this step, the team must prioritize the requirements taking into account factors such as cost, time to implement, and severity. Prioritizing gives a schedule of which requirements should be developed and in which order.

9. **Requirements Inspection** – The final step is to validate the requirements and ensure that there are no vulnerabilities or defects in the requirements. This step provides teams one final chance to verify and correct any mistakes.

Performing the nine steps of the SQUARE process can be a long process, thus a shortened variation known as SQUARE-Lite [2] was tested. This shortened version incorporates the following five steps: agree on definitions, identify security goals, perform risk assessment, elicit security requirements, and prioritize requirements. With the fewer number of steps, SQUARE-Lite is able to reduce the overall time required to complete the process.

2.2. Privacy SQUARE Methodology

SQUARE has been recently adapted for usage with privacy requirements engineering [3]. Privacy requirements engineering poses a range of different challenges compared to security requirements engineering. Privacy policies of an application are not dictated solely by the stakeholders, but they can be mandated by laws and regulations. Laws and regulations can be difficult to understand; a fair amount of knowledge may be required to properly interpret and comply with all applicable laws. Goal-Based Requirements Analysis Method, Pattern-Based Approach, and E-Commerce Personalization Approach have all been used in privacy requirements engineering; however, these methods are generic in nature and require a detailed understanding of privacy laws, standards and policies [3].

This adaptation of SQUARE aims at reducing the difficulty and expertise required to properly perform privacy requirements engineering. Many of the steps in SQUARE for privacy
are the same as those for security, but minor differences do exist. Privacy requirements are aimed at protecting specific pieces of information (e.g. social security numbers and medical history information). In the second step for privacy, requirements engineers must identify assets that are to be protected in addition to privacy goals.

2.3. mySQUARE

mySQUARE [4] was the first tool to provide semi-automation for SQUARE for security requirements engineering. It is a desktop application and can only be used by one person at a time. mySQUARE provides guidance information as a user works through the steps and at the end of the process, the tool is able to generate reports that would be fairly labor-intensive to perform manually. mySQUARE also has the capability to export all of the data into an xml formatted file, which can be used with other tools.

This application was able to reduce the workload and expertise required to use the process but it was limited to how many users could use it and did not have a built in risk assessment tool. SQUARE requires a certain amount of communication between requirements engineers and stakeholders, an application restricted to one user means collaboration is performed outside of the tool.

2.4. CyLab SQUARE Tool

CyLab, in cooperation with the SEI, developed the CyLab SQUARE tool [5]. This tool is a web-based tool that allows for teams of any size to work concurrently on a project. This tool also implemented each step using the recommended methods suggested in the SQUARE documentation [1]. The CyLab SQUARE tool allows teams to manage the SQUARE process from beginning to end without the need to use external applications, and it provides guidance so that project teams with limited experience in security are still able to perform SQUARE.
However, this tool is limited to security requirements engineering and only provides a single interface for risk assessment and requirements elicitation.

Several papers have indicated [3, 6] that a new tool called P-SQUARE is being developed to handle privacy requirements engineering. Outside of being mentioned in the future works sections of several papers, there is no documentation that details its future functionality at this time.

2.5. eSQUARE

eSquare [7] is another web based tool that provides similar functionality as the CyLab tool and was developed using Java Server Pages (JSP). It provides support which allows multiple users to work on a project and gives guidance on using SQUARE. eSQUARE provides interfaces for performing risk assessment and requirements elicitation. It also includes support for the formal specification language Z. This provides a facility for taking in a formal specification of security requirements, verifying them for correctness and completeness.

2.6. 2-SQUARE

All of the previously mentioned tools provide support for the original goal of SQUARE, security requirements engineering. Each of the tools has limited or single-method support for processes for performing risk assessment and requirements elicitation. 2-SQUARE is a web-based tool for semi-automation of both security and privacy, and it provides flexibility for requirements engineering teams using SQUARE. 2-SQUARE is designed to be highly expandable to allow for the addition of new variations of SQUARE, risk assessment, and requirements elicitation modules.
2-SQUARE has generic methods for inputting risk assessments or requirements, but it also includes modules for the recommended methods for risk assessment and requirements elicitation. Modules preprogrammed include National Institute of Standards and Technology (NIST) 800-30 guide [8], Privacy Risk Analysis for Ubiquitous Computing [6] and Privacy Requirements Elicitation Technique (limited) [9].
Chapter 3

2-SQUARE DESIGN

3.1. Functional and Non-Functional Requirements

2-SQUARE is making the process of security and privacy requirements engineering easier by semi-automating the SQUARE process. The functional requirements aim at organizing the artifacts produced during each step of the SQUARE process as well as providing help where needed. 2-SQUARE includes the following functionality: workflows driven by database values, ease of maintaining workflows, integrated help and suggestions, data entry for process artifacts, ability to upload files, keep notes, and storage of documents. Risk assessment and requirements elicitation steps are also provided as expandable modules such that new methods can be incorporated without having to rewrite the entire application.

There are currently no single requirements engineering techniques that can satisfy how project teams work, thus 2-SQUARE heavily took into consideration the nonfunctional requirements of expandability and maintainability in the architectural design. 2-SQUARE has generic methods built in for the risk assessment and requirements elicitation steps, but they can be easily expanded through modules to support specific methods such as the National Institute of Standards and Technology 800-30 and Privacy Risk Analysis for Ubiquitous Computing specifications for risk assessment, which is included in the current version.

The 2-SQUARE application had to be usable and available to allow for an easy learning curve. Usability was the driving force for the design of the user interface pages providing a consistent and straight forward user interface. Help with the SQUARE process and guidance with the interface is available in the navigational bar on all pages.
3.2. 2-Square Architecture Design

2-SQUARE is designed as a web application so that it can be accessed by multiple people from anywhere and provides for an easy method for updating the SQUARE method and module expansion. 2-SQUARE’s design pattern makes it easier for a programmer to add a new variation of SQUARE or create new modules to support risk assessment or requirements elicitation modules.

The MVC pattern was chosen for 2-SQUARE because of the separation of concerns inherit in the design pattern. This separation of concerns keeps separate the code for the data models, user interface, and the business rules, and it reduces the amount of code coupling. This attribute provides for a more maintainable and easier to understand code base, allowing for the ease of expansion of new modules and even new variations of the SQUARE method. The ASP.NET MVC framework combined with the ADO.NET Entity Framework (EF) provide a very powerful platform for creating web applications that provide security, ease of use and rapid development. These two frameworks allow for a reduction in the number of layers required for development. Generally, in a web-based application they require 4 layers of abstraction: presentation, business, data access layers, and the database. The Entity Framework provides an object relational mapping (ORM) of the data models to the database. The EF framework takes the data models and automatically generates the database with all of the necessary constraints, thus eliminating the need to separately handle the database. The basic architecture of the interactions between the model view controller, workflow, and entity framework is shown in Figure 1.
The SQUARE workflow, as shown in Figure 2, was designed to operate as a service that the tool’s controllers can call to execute data changes and validation against business logic. The services are designed to interact with the controllers and the entity framework. The workflow is divided into two separate services: one that checks and updates the workflow, and another that provides all of the Create, Read, Update, and Delete (CRUD) operations to the database.
The user interface is implemented in the View layer using standard HTML and CSS. Using the Asp.Net MVC view engine provides the benefits of built-in support for defending against common web security vulnerabilities (e.g., cross site scripting) as well as easy to understand syntax for rendering server side models. All pages inherit their overall design from a master page to ensure a consistent look and feel. All of the guidance information was taken from the SQUARE documentation as well as from the SEI website [1, 3, 10].
The data models are divided into four categories based on their functions in the system. The first category of models is used to drive the SQUARE workflow, providing information on security and privacy steps. These models contain information on guidance and navigation. The second category of models is the objects used to store all project information and artifacts. These objects are used throughout the workflow and in the generated reports. The third set of models is used for risk assessment and elicitation technique modules. These objects are used specifically for modules and are ignored by the primary SQUARE workflow. The end result of the modules output data is converted into the standard artifact tables. These three categories are each mapped to database tables where the ORM matches them up to the generated tables. The final category of models is known as view models and these are only used to provide views with the necessary information to render the user pages.

The workflow is driven in the controller layer by using information contained in the process management object models. Leveraging the models allows for a more generic architecture in the main tool that is associated with the steps for SQUARE. This allows for sharing of common code between SQUARE variations where appropriate.

3.3. 2-Square User Interface Design

2-SQUARE’s screens are designed with the non-functional requirement usability in mind; the screens are simple, easy to use, and consistent. Every page in the tool inherits its main design from a master design page; this ensures that the header and navigation remains consistent throughout the tool. The navigation bar (as seen in Figure 3) shows the navigational bar customized for the categorization step. The tool has two basic designs available to the user for the majority of the page. One of the designs displays the project information, such as terms and definitions in a table format. The other type of page design provides a form for users to enter
information such as artifacts. This basic design simplifies the application, making for an easier learning curve.

Figure 3. 2-SQUARE Navigational Bar, Categorization Step

Figure 4. Authenticated User Home Page

The tool allows users to work on multiple projects, thus the home page (as seen in Figure 4) displays a list of projects that the user has access to. On this screen, the user can select an existing project to work on or create a new project.
Once a project has been selected, the user is taken to the project home page which contains a list of the steps for the selected project. Figure 5 shows a partial list of steps for an in progress project. Steps are color coded and given distinct icons to let the user know the working progress of the project. Red and check mark signify a complete status, green and an arrow let the user know that step is in progress and yellow and a minus in a circle indicate a pending status. Users click on a specific step to be taken to the page for that step and administrators are also able to go to a page for updating project step status.

Throughout the project step pages, the shared notes and files can be activated from buttons that are present on the bottom of the page as circled in Figure 6. Pressing the “Project Step Notes” button presents the dialog containing notes from all team members and the ability to add more notes. The “Project Step Files” makes another dialog appear with a list of files that are
being shared to help complete the current task, ex. a diagram may be necessary for understanding a security goal, but it is not necessary for the final documentation.

Figure 6. Notes and File Sharing Functions

As shown in Figure 7, the list of selected terms and definitions are displayed to the user. From this page, the user has the option to edit existing terms, add a new term or select to add a term from a predefined list of common security terms. The list shows the selected terms along with the source and definition.
Security Step 1 - Agree on Definitions

Selected Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Source</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>Wikipedia</td>
<td>a system which enables an authority to control access to areas and resources in a computer-based information system</td>
</tr>
<tr>
<td>control</td>
<td>Wikipedia</td>
<td>a system which enables an authority to control access to areas and resources in a computer-based information system</td>
</tr>
</tbody>
</table>

Figure 7. List of Selected Definitions Page

Business Goal

Business goal to be defined.

Save

Security Goals

<table>
<thead>
<tr>
<th>Security Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
</tr>
<tr>
<td>Edit</td>
</tr>
</tbody>
</table>

Figure 8. Business and Security Goals Page
The identification of goals for SQUARE’s third step differs slightly between the security and privacy variations, which are highlighted in Figure 8 and Figure 9, respectively. In the security page, the user is able to enter one business goal and add, edit, and delete security goals. In contrast, the privacy page allows users to enter multiple assets and privacy goals. In order to enter an asset or goal, the user enters this information using a simple form.
Figure 10 represents the third step of SQUARE – the collection of artifacts. All artifacts such as architecture diagrams, use/misuse cases, and attack trees can be uploaded into 2-SQUARE. The list of current documents is presented to the user with the option to edit, view, delete, or add new artifacts.

The fourth step of SQUARE, risk assessment, stands out from the previous steps because it is handled through modules. The workflow for individual modules can be very different from that of the core SQUARE functions and may dictate a specific design. Thus, modules may or may not follow the consistent design.

<table>
<thead>
<tr>
<th>Security Step 6 - Elicit Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
</tr>
<tr>
<td>Edit</td>
</tr>
<tr>
<td>Edit</td>
</tr>
<tr>
<td>Edit</td>
</tr>
</tbody>
</table>

Figure 11. Security Requirements List Page

In step five, the user is presented with a list of various elicitation techniques along with strengths and weaknesses of each method. The list that is presented can be updated with new methods as new modules for requirements elicitation are developed. Selecting a technique requires a justification to be entered to explain to the team the reasoning for selecting the particular methodology. After a technique is selected the user is either presented with a custom module for the selected technique, or if one doesn’t exist, the generic module is presented. Figure
Figure 12. Requirements Categorization Page
Figure 12 shows the categorization screen, which gives the interface for step seven of the process. The screen is divided into two lists, the first of which is the categorized requirements followed by the uncategorized requirements. There is an option in the navigation bar that allows
users to manage the list of categories. The categories are then used in the form used to categorize requirements.

Step eight, which is illustrated in Figure 13, shows the screen to prioritize requirements. This screen provides a list of all the requirements with the associated category. There are multiple methods for entering priority: users can enter a value in the priority column, drag and drop a requirement or perform a combination of the two.

Figure 14 presents the screen for the ninth step of the SQUARE process. It shows the requirements with the category and is ordered by priority. Users are able to review all of the requirements and add any concerns as a defect in the requirement for review and modification. Once a defect has been corrected, users are able to mark it as resolved.
Chapter 4

2-SQUARE IMPLEMENTATION

The tool was developed using iterations, starting with security and shared objects to setup a foundation for a secure application. Then with successive iterations each of the nine steps were developed for security and privacy along with the necessary service methods. In this manner the tool was verified for each step as they were being developed.

4.1. View Implementation

The views are the web pages that provide an interface for the user to perform their functions. The views are written using Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, and syntax specific to the ASP.NET MVC view engine. The MVC view engine allows for dynamic page context for elements such as the navigation bar.

The view engine is able to take view models and render the interactive screens to the users. Data is rendered to the screens using standard HTML elements and JavaScript is used to enhance the interactive experience for a fluid interface. A limited amount of input validation is performed on the pages before any communication with the server occurs to provide the users with rich feedback. These screens are then able to communicate with the server using HTML form posts and asynchronous JavaScript and XML (AJAX) through actions within the controllers.

4.2. Controller Implementation

Controllers are C# classes that are used to handle the requests that come from the users. They are broken down into two types: one type is used for managing the workflow, and the other is responsible for handling the creation and editing of data objects. All controllers inherit from a base application controller that provides basic objects to be used across each of the controllers.
As seen in Figure 15, the SQUARE process controllers inherit from the application controller and a SQUARE type interface. The interface is used to ensure that the process controllers implement the necessary steps and have the parameters needed to work with step specific methods that are shared. The SQUARE process is driven by a combination of database values which define a variation to SQUARE (privacy or security) and the interpretation of those database values in the tool. This allows for the addition of SQUARE variations (ex. SQUARE-Lite) without the need to rewrite entire sections of the tool by simply adding in additional controllers that are able to function independently.
The other types of controllers are responsible for the handling of database objects specific to projects such as definitions, artifacts and requirements. These controllers perform the interaction between view layer and the services and are responsible for rendering the page forms and saving that information to the database. There are two different types of these controllers, as seen in Figure 16: those that handle project artifacts, and those responsible for modules. Since the modules can be added by any programmer, they inherit from an interface to enforce certain parameters, which help ensure compatibility with the core application.
4.3. Services Implementation

The services are responsible for executing the business logic and ensuring that the inputs are valid. These services are given to the controllers through a technique called dependency injection. This method ensures that all dependent components are available to the controllers so that they may execute their functions. The services validate the input not only for the object integrity, but they also ensure that objects are only being written for active steps.

Interaction between the services and the database are executed using Microsoft’s Entity Framework (EF). Use of the entity framework includes access to the Language Integrated Query (LINQ) extensions. LINQ is a language very similar to Structured Query Language (SQL), but it makes complex queries easier through member traversal as opposed to joining tables. As a result of this dependency on data models, queries can be validated at compile time.

4.4. Data Model and Database Implementation

All of the data objects are modeled using C# classes, and they are automatically mapped into the database using EF into a 1-1 relationship. This reduces the need to manage the database independently as any changes made to the data models would propagate the change to the database automatically. The data models were designed to follow database normalization rules by having repeating data stored in separate tables. This allows for smaller and more efficient tables as well as the ability to populate drop-down lists from the database.
The domain objects, and in turn the database, is divided into three categories of objects/tables based on their usage. The first set of objects is used to control the workflow; these tables are responsible for holding definitions of the variations of SQUARE. As seen in Figure 17, these tables hold the steps and limited routing information used by the tool to dynamically generate menu screens and navigational buttons. The second type of objects is used for storage of all artifacts specific to a project and the data for the SQUARE process. The information stored includes artifacts and the final list of requirements. The final set of objects includes those related to specific technique modules and can be used by the techniques to store intermediate
information. Since 2-SQUARE is not concerned with the intermediate information, this data is stored separately. At the end of using a module, the data is transformed and moved into the SQUARE tables. For example, when performing risk assessment, different calculations and intermediate steps may need to be performed, but 2-SQUARE only needs to know the end result.
There are no complete case studies that fully implement SQUARE in its current form; however, there is one for security requirements engineering [11] and another case studies in privacy risk assessment [6]. Because the majority of steps are similar for both security and privacy, we are working under the assumption that benefits gained in security would translate to similar benefits for privacy. We will cover the first case study on security, and then we will discuss the risk assessment case study for privacy.

In the early stages of the development of the SQUARE method, six graduate students worked under the direction of Professor Nancy Mead and tested SQUARE with the Acme Company on a product called Asset Management System (AMS) [11]. The Acme Company is a relatively small organization with 1,000 employees nationwide that develops several products. AMS is a tool that provides management of critical IT assets and provides decision support capabilities. Acme is planning the development of a new version of AMS; however, there is little to no documentation for the existing system.

The case study team’s methodology for testing SQUARE was to perform the tasks, receive feedback from the client, and incorporate their feedback to improve the system documents and the process. The project team took the role of external consultants and worked with the technical lead and assistant within the company. The project team followed the SQUARE steps as closely as possible, but ultimately found that minor modifications to the steps were necessary for this particular case. Although the team did not follow SQUARE steps exactly, they did note
that every application of SQUARE is likely to require slight changes. After taking feedback from the clients into consideration, the SQUARE process was updated.

2-SQUARE was used on the outputs generated from the aforementioned case study to show the effectiveness that this tool has in making it easier to apply SQUARE towards security requirements engineering. Security artifacts generated from the case study were entered into the system. 2-SQUARE eliminates the need for project teams to pass documents back and forth, and it allows for comments to be integrated into the steps.

![2-Square](image)

**Security Step 1 - Agree on Definitions**

<table>
<thead>
<tr>
<th>Term</th>
<th>Source</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>access control</td>
<td>[SANS Institute]</td>
<td>Access control ensures that resources are only granted to those users who are entitled to them.</td>
</tr>
<tr>
<td>access control list</td>
<td>[<a href="http://watis.techtarget.com">http://watis.techtarget.com</a>]</td>
<td>A table that tells a computer operating system which access rights or explicit denials each user has to a particular system object, such as a file directory or individual file.</td>
</tr>
<tr>
<td>antivirus software</td>
<td>[<a href="http://watis.techtarget.com">http://watis.techtarget.com</a>]</td>
<td>A program that searches hard drives and floppy disks for any known or potential viruses.</td>
</tr>
<tr>
<td>artifact</td>
<td>[Handbook for Computer Security Incident Response Teams]</td>
<td>The remnants of an intruder attack or incident activity. These could be software used by intruder(s), a collection of tools, malicious code, logs, files, output from tools, or the status of a system after an attack or intrusion.</td>
</tr>
</tbody>
</table>

Figure 18. AMS Selected Definitions

A set of common security and privacy terms and definitions is preloaded in the database—many of which were loaded based on the results of the AMS case study and the original paper
that outlines SQUARE. Selecting an initial list of terms and definitions as seen in Figure 18, was simple given the common set of terms and then adding specific ones to this project.

Because of the lack of documentation that existed for AMS, the project team had to generate several diagrams out of step in order to get a better understanding of the system. These types of documents were uploaded into the system to enable better sharing and documentation as seen in Figure 19. These documents were used only to facilitate the identification of business and security goals and are not included in any documentation. Security and safety goals were then entered into 2-SQUARE. All the remaining documents generated during the fourth step were uploaded into the program. These diagrams included, but were not limited to, use case diagrams, use cases in tabular format, and misuse case documents.
Figure 20. Entering First AMS Requirement

Requirements and prioritizations were derived from the misuse cases defined earlier in the process. Based on the severity and risk of the misuse cases, they were given a priority scaled from 1-10 with 10 being high. The priority values were then broken down to low, medium and high. The method used by the project team to derive requirements and prioritizations is not directly supported by 2-SQUARE; however, the resulting requirements generated from the method used were entered into 2-SQUARE through the generic module that was included, as seen in Figure 20.
2-SQUARE provides two different ways of signifying the priority of a requirement: either by changing the order of requirements or placing a value in a priority box. We used the priority box with the following values: 1 for low, 2 for medium, and 3 for high. In the case study, the team prioritized the misuse cases that the requirements were derived from. Because many of the requirements were the result of several misuse cases, many of the requirements received a high priority, as seen in Figure 21.

Figure 21. Prioritization AMS Requirements
The final step of SQUARE requires the project team to verify the security requirements and correct any defects. Figure 22 presents all of the developed requirements along with their respective categories and priority levels, and it provides a method for team members to flag defects with the requirements so that they can be reevaluated and resolved. Once the requirements have been accepted, the documents can be generated and the security requirements engineering process is complete.
Figure 23. Risks in Privacy Risk Analysis for Ubiquitous Computing Module

The second case study concentrated on risk assessment techniques with respect to privacy. This case study analyzed a tool used for a research-oriented university to manage awards received for research activities [6]. The application of this risk assessment case study utilized the Privacy Risk Analysis for Ubiquitous Computing [6] method, which does have a module built into 2-SQUARE. Based on the results from an interview with the stakeholders, the identified risks were entered into the risk assessment module and given rankings of likelihood and damage and cost. 2-SQUARE’s module, pictured in Figure 23, was then able to determine if a protection should be considered for each specified risk, and it allows for entering recommendations on how to deal with each identified risk.
Chapter 6

CONCLUSION

SQUARE is a structured and highly adaptable requirement engineering methodology that simplifies and effectively performs security and privacy requirements engineering. 2-SQUARE builds on SQUARE and provides users with a tool that can aid with utilizing the SQUARE process as well as facilitating communications between requirements engineers and stakeholders. Users are able to enter developed artifacts related to the project and generate documentation at the end of the process. The ability to upload documents and pass comments between users reduces the need to communicate between users outside of the system and easily provides users with up to date information on artifacts. Because SQUARE allows project teams to apply different processes to perform risk assessment and requirements elicitation, 2-SQUARE also provides that type of flexibility while allowing teams to select or even create processes that work best for themselves.

Although 2-SQUARE was designed to be able to handle a wide variety of situations, the future addition of more risk assessment and requirements elicitation modules would greatly increase its usefulness for more project teams. Because the set of preloaded terms and definitions can become stale, keeping these up to date will ensure the usefulness of the features built in for the first step. Finally because 2-SQUARE is able to handle additional SQUARE variations, the development of a SQUARE-Lite variation would make 2-SQUARE a complete SQUARE tool without the need to develop an entirely new tool.
APPENDIX

2-SQUARE Requirements Document

Asset Management System

**Description** IT Management system, developed for the Acme Company as part of a case study for the SQUARE process. More details can be obtained at the following addresses.
(http://www.sei.cmu.edu/library/abstracts/reports/04sr015.cfm)
(http://www.sei.cmu.edu/library/abstracts/reports/05sr005.cfm)
(http://www.sei.cmu.edu/library/abstracts/reports/06sr003.cfm)

Security

**Terms**

**access control** - Access control ensures that resources are only granted to those users who are entitled to them. [SANS Institute]

**access control list** - A table that tells a computer operating system which access rights or explicit denials each user has to a particular system object, such as a file directory or individual file [http://watis.techtarget.com]

**antivirus software** - A program that searches hard drives and floppy disks for any known or potential viruses [http://watis.techtarget.com]

**artifact** - The remnants of an intruder attack or incident activity. These could be software used by intruder(s), a collection of tools, malicious code, logs, files, output from tools, or the status of a system after an attack or intrusion [Handbook for Computer Security Incident Response Teams]

**asset** - A critical valuable that a company owns and wants to secure. [SQUARE Case Study]

**attack** - An action conducted by an adversary, the attacker, on a potential victim. A set of events that an observer believes to have information assurance consequences on some entity, the target of the attack [A Trustworthy Refinement Through Intrusion-Aware Design]

**auditing** - The information gathering and analysis of assets to ensure such things as policy compliance and security from vulnerabilities [http://www.sans.org/resources/glossary.php]

**authentication** - The process of determining whether someone or something is, in fact, who or what it is declared to be [http://watis.techtarget.com]

**availability** - The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications
for the system; i.e., a system is available if it provides services according to the system
de- sign whenever users request them

back door - An element in a system that allows access by bypassing access controls
breach - Any intentional event in which an intruder gains access that compromises the confi-
dentiality, integrity, or availability of computers, networks, or the data residing on them
brute force - A cryptanalysis technique or other kind of attack method involving an
exhaustive procedure that tries all possibilities, one by one
buffer overflow - A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information—which has to go somewhere—can overflow into adjacent buffers, corrupting or overwriting the valid data held in them

buffer over- flow - A buffer overflow occurs when a program or process tries to store
more data in a buffer (temporary data storage area) than it was intended to hold. Since
buffers are created to contain a finite amount of data, the extra information—which has
to go somewhere—can overflow into adjacent buffers, corrupting or overwriting the
valid data held in them


cache cramming - The technique of tricking a browser to run cached Java code from the
local disk instead of the Internet zone, so it runs with less restrictive permissions

cache poisoning - Malicious or misleading data from a remote name server is saved

confidentiality - The property that information is not made available or disclosed to
unauthorized individuals, entities, or processes (i.e., to any unauthorized system entity)

control - An action, device, procedure, or technique that removes or reduces a
vulnerability

corruption - A threat action that undesirably alters system operation by adversely
modifying system functions or data

cracker - Someone who breaks into someone else’s computer system, often on a
network; by-passes passwords or licenses in computer programs; or in other ways
intentionally breaches computer security

denial-of-service (DoS) attack - A form of attacking another computer or company by
sending millions of requests every second, causing the network to slow down, cause
errors, or shut down

disaster recovery plan - A disaster recovery plan (DRP)—sometimes referred to as a
business continuity plan (BCP) or business process contingency plan (BPCP)—describes
how an organization is to deal with potential disasters

disclosure - The dissemination of information to anyone who is not authorized to access
that in- formation

disgruntled employee - A person in an organization who deliberately abuses or misuses
computer systems and their information

disruption - A circumstance or event that interrupts or prevents the correct operation of
system services and functions

- **downtime** - The amount of time a system is down in a given period. This will include crashes and system problems as well as scheduled maintenance work

- **encryption** - Cryptographic transformation of data (called “plaintext”) into a form (called “cipher text”) that conceals the data’s original meaning to prevent it from being known or used

- **espionage** - The act or practice of spying or of using spies to obtain secret information about another government or a business competitor

- **essential services** - Services to users of a system that must be provided even in the presence of intrusion, failure, or accident

- **exposure** - The dissemination of information to anyone who is not authorized to access that information

- **fabrication** - Aims to fool other machines on the network into accepting the imposter as an original, either to lure the other machines into sending it data or to allow it to alter data

- **fault line attacks** - Fault line attacks use weaknesses between interfaces of systems to exploit gaps in coverage

- **fault tolerance** - Describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can immediately take its place with no loss of service. Fault tolerance can be provided with software, or embedded in hardware, or provided by some combination

- **firewall** - A system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both

- **hacker** - An individual who breaks into computers primarily for the challenge and status of obtaining access

- **honey pot** - Programs that simulate one or more network services designated on a computer’s ports. An attacker assumes that vulnerable services that can be used to break into the machine are being run. A honey pot can be used to log access attempts to those ports, including the attacker’s keystrokes. This can provide advanced warning of a more concerted attack

- **HTTP header manipulation** - HTTP requests and responses send information in the HTTP headers. HTTP headers are a series of lines containing a name/value pair used to pass information such as the host, referrer, user agent, etc. HTTP headers can be manipulated to cause SQL injection or cross-site scripting errors.

- **impact** - The negative effect of an attack on a victim system by an attacker

- **incident** - An adverse network event in an information system or network or the threat of the occurrence of such an event
incident handling - An action plan for dealing with intrusions, cyber theft, denial of service, fire, floods, and other security-related events  
[[http://www.sans.org/resources/glossary.php]]

insider threat - The threat that authorized personnel of an organization will act counter to the organization’s security and interest, especially for the purposes of sabotage and espionage  
[[http://www.hpcc-usa.org/pics/02-pres/wright.ppt]]

integrity - For systems, the quality that a system has when it can perform its intended function in a unimpaired manner, free from deliberate or inadvertent unauthorized manipulation. For data, the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner  

interception - Access to an asset gained by an unauthorized party  
[Security in Computing]

intrusion - An attack on a network for the purpose of gaining access to or destroying privileged information or disrupting services to legitimate users  
[A Trustworthy Refinement Through Intrusion-Aware Design]

intrusion detection system - A combination of hardware and software that monitors and collects system and network information and analyzes it to determine if an attack or an intrusion has occurred. Some ID systems can automatically respond to an intrusion  

intrusion prevention system - A system used to actively drop packets of data or disconnect connections that contain unauthorized data. Intrusion prevention technology is also commonly an extension of intrusion detection technology  
[www.wikipedia.org]

liability - The responsibility of someone for damage or loss  
[[http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html]]

luring attack - A type of elevation of privilege attack where the attacker “lures” a more highly privileged component to do something on his or her behalf. The most straightforward technique is to convince the target to run the attacker’s code in a more privileged security context  
[“Item 7: What is a Luring Attack?” The .NET Developer’s Guide to Windows Security]

malware - Programming or files that are developed for the purpose of doing harm. Thus, malware includes computer viruses, worms, and Trojan horses  
[[http://www.webopedia.com]]

man-in-the-middle attack - An attack in which the attacker is able to read, and possibly modify at will, messages between two parties without letting either party know that they have been attacked. The attacker must be able to observe and intercept messages going between the two victims  
[[http://encyclopedia.thefreedictionary.com/man%20in%20the%20middle%20attack]]

masquerade - Aims to fool other machines on the network into accepting the imposter as an original, either to lure the other machines into sending it data or to allow it to alter data  
[[http://www.cert.org/research/taxonomy_988667.pdf]]

modification - Situation in which an unauthorized party not only gains access to, but
tampers with an asset [http://www.cert.org/research/JHThesis/Start.html]

**non-essential services** - Services to users of a system that can be temporarily suspended to permit delivery of essential services while the system is dealing with intrusions and compromises [http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html]

**non-repudiation** - The goal of non-repudiation is to prove that a message has been sent and received [http://www.ssimail.com/Glossary.htm#N]

**patch** - A small update released by a software manufacturer to fix bugs in an existing program [http://www.sans.org/resources/glossary.php]

**patching** - The process of updating software to a new version that fixes bugs in a previous version [http://www.sans.org/resources/glossary.php]

**penetration** - Intrusion, trespassing, or unauthorized entry into a system [http://www.yourwindow.to/information-security]

**penetration testing** - The execution of a testing plan, the sole purpose of which is to attempt to hack into a system using known tools and techniques [http://www.yourwindow.to/information-security]

**physical security** - Security measures taken to protect systems, buildings, and related supporting infrastructure against threats associated with their physical environment [http://encyclopedia.thefreedictionary.com/man%20in%20the%20middle%20attack]

**port scanning** - The act of systematically scanning a computer’s ports [http://www.webopedia.com]

**privacy** - The quality or condition of being secluded from the presence or view of others [http://dictionary.reference.com]

**procedure** - The implementation of a policy in the forms of workflows, orders, or mechanisms [http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html]

**recognition** - The capability of a system to recognize attacks or the probing that precedes attacks [A Trustworthy Refinement Through Intrusion-Aware Design]

**recovery** - A system’s ability to restore services after an intrusion has occurred. Recovery also contributes to a system’s ability to maintain essential services during intrusion [A Trustworthy Refinement Through Intrusion-Aware Design]

**replay attack** - The interception of communications, such as an authentication communication, and subsequent impersonation of the sender by retransmitting the intercepted communication [http://www.ffiec.gov/ffiecinfo/Booklets/information_security/08_glossary.html]

**resilience** - The ability of a computer or system to both withstand a range of load fluctuations and also remain stable under continuous and/or adverse conditions [http://www.yourwindow.to/information-security]

**resistance** - Capability of a system to resist attacks [A Trustworthy Refinement Through Intrusion-Aware Design]

**risk** - The product of the level of threat with the level of vulnerability. It establishes the likelihood of a successful attack [http://www.sans.org/resources/glossary.php]

**risk assessment** - The process by which risks are identified and the impact of those risks determined [http://www.sans.org/resources/glossary.php]
script kiddies - The more immature but unfortunately often just as dangerous exploiter of security lapses on the Internet. The typical script kiddy uses existing and frequently well known and easy-to-find techniques and programs or scripts to search for and exploit weaknesses in other computers on the Internet—often randomly and with little regard or perhaps even understanding of the potentially harmful consequences [http://watis.techtarget.com]

security policy - A policy that addresses security issues [http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html]

spoof - The term is used to describe a variety of ways in which hardware and software can be fooled. IP spoofing, for example, involves trickery that makes a message appear as if it came from an authorized IP address [http://www.webopedia.com]

SQL injection - A type of input validation attack specific to database-driven applications where SQL code is inserted into application queries to manipulate the database [http://www.sans.org/resources/glossary.php]

stakeholder - Anyone who is a direct user, indirect user, manager of users, senior manager, operations staff member, support (help desk) staff member, developer working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a soft- ware project [http://www.agilemodeling.com/essays/activeStakeholderParticipation.htm]

stealthing - A term that refers to approaches used by malicious code to conceal its presence on an infected system [http://www.sans.org/resources/glossary.php]

survivability - The capability of a system to complete its mission in a timely manner, even if significant portions are compromised by attack or accident. The system should provide essential services in the presence of successful intrusion and recover compromised services in a timely manner after intrusion occurs [http://www.sei.cmu.edu/publications/documents/03.reports/03tm013.html]

target - The object of an attack, especially host, computer, network, system, site, person, organization, nation, company, government, or other group [http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html]

threat - A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm [http://www.sans.org/resources/glossary.php]

threat assessment - The identification of the types of threats that an organization might be exposed to [http://www.sans.org/resources/glossary.php]

threat model - Used to describe a given threat and the harm it could to do a system if it has a vulner- ability [http://www.sans.org/resources/glossary.php]

toolkits - A collection of tools with related purposes or functions, e.g., antivirus toolkit, disk toolkit [http://www.yourwindow.to/information-security]

Trojan - A program in which malicious or harmful code is contained inside apparently harm- less programming or data in such a way that it can get control and do its chosen form of damage, such as ruining the file allocation table on a hard disk [http://watis.techtarget.com]
trust - Determines which permissions other systems or users have and what actions they can perform on remote machines [http://www.sans.org/resources/glossary.php]

uptime - The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them [http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html]

victim - That which is the target of an attack. An entity may be a victim of either a successful or unsuccessful attack [http://www.sans.org/resources/glossary.php]

virus - A hidden, self-replicating section of computer software, usually malicious logic, that propagates by infecting—i.e., inserting a copy of itself into and becoming part of—an other program. A virus cannot run by itself; it requires that its host program be run to make it active [http://www.sans.org/resources/glossary.php]

vulnerability - A condition or weakness in (or absence of) security procedures, technical controls, physical controls, or other controls that could be exploited by a threat [http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf]

worm - A self-replicating virus that does not alter files but resides in active memory and duplicates itself. Worms use parts of an operating system that are automatic and usually invisible to the user. It is common for worms to be noticed only when their uncontrolled replication consumes system resources, slowing or halting other tasks [http://watis.techtarget.com]

Business Goal
This tool is not intended to provide canned responses to every possible scenario, but instead provides the means to make informative decisions based on available sources.

Security Goals
-Confidentiality
Insure that information and resources are accessed only by those who are authorized to use them

-Availability
Ensure that the Asset Management System is functional and available at all times. This includes the core facility management services, Sybase databases, etc.

-Data Integrity
Integrity of data is absolutely critical in the Asset Management System package. Data backups and checksum integrity verification are of the utmost importance

-Monitoring
Preserve or enhance the ability to accurately record the activities that take place.

-Access Control
Only authorized users of the Asset Management System have access to their specified and permissible resources

-Maintaining Mission Critical Services
The most important goal is the ability to deliver essential services in the face of attack,
failure, or accident.

**Disaster Recovery**
Have a current disaster recovery plan in the event of an emergency or service disruption.

**Code Review**
Periodic code review should be performed to ensure script and code confidentiality and integrity

**Elicitation Technique** Not Listed
**Rationale** This was the selected method used by the project team.

**Requirements**

<table>
<thead>
<tr>
<th>Id</th>
<th>Requirement</th>
<th>Category</th>
<th>Priority</th>
<th>Essential</th>
</tr>
</thead>
<tbody>
<tr>
<td>SU-4</td>
<td>The operating system, applications, firewalls, and IDS must be patched routinely.</td>
<td>Survivability</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-9</td>
<td>Follow the principle of least privilege and use least privileged service account to run processes and access resources.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-5</td>
<td>User activities must be periodically reviewed</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-1</td>
<td>The system shall audit systems on network and user logging information. This shall be put into practice monthly.</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PV-6</td>
<td>Require users to change their passwords periodically.</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AC-5</td>
<td>Users should not have rights or access levels beyond those which are prescribed by their job responsibilities.</td>
<td>Access Control</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AC-2</td>
<td>The system shall not grant any client application access to one or more system</td>
<td>Access Control</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AN-2</td>
<td>Authentication control mechanism shall be enforced in production environment. Authentication control will be done on the network level.</td>
<td>Authentication</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-7</td>
<td>A DBA and/or manager performs information integrity checks on a routine basis.</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PV-1</td>
<td>The system shall not permit any user login data to be retrieved by an attacker.</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SU-6</td>
<td>Do not set up shared files/folders/drives on the network.</td>
<td>Survivability</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SU-5</td>
<td>New systems on the network should be evaluated prior to deployment.</td>
<td>Survivability</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AC-1</td>
<td>The system shall not grant any user access to one or more system services and data during a secure session without first identifying the user.</td>
<td>Access Control</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AN-1</td>
<td>Authentication control mechanism shall be</td>
<td>Authentication</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PV-3</td>
<td>Password-protect any necessary shared documentation</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------</td>
<td>---------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>PV-5</td>
<td>Users should not reveal their account names and passwords in any situation</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AC-4</td>
<td>Set clear and defined user access control for all users (Low, Medium, High, System Admin).</td>
<td>Access Control</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-10</td>
<td>Perform routine code review</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SU-1</td>
<td>The system shall continue to fulfill its mission in the presence of an attack (possibly in minim and safe mode).</td>
<td>Survivability</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-2</td>
<td>The system shall be able to determine when a buffer overflow attack has occurred. If attack takes place, system should shut down and system administrator should be notified within a reasonable time.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-6</td>
<td>Configuration changes are stored and crossreviewed.</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-1</td>
<td>The system shall protect itself from viruses by using virus detection software with updated signatures. The virus detection software should also be run weekly.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-5</td>
<td>The system shall protect itself from input validation attack.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-3</td>
<td>Log all incoming and outgoing traffic.</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-4</td>
<td>The system shall protect itself from attacks due to weak server configuration.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DR-1</td>
<td>Develop disaster recovery and contingency plan.</td>
<td>Disaster Recovery</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DR-2</td>
<td>Perform routine system and data backup.</td>
<td>Disaster Recovery</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-4</td>
<td>Separate personnel review system administrator activities</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-3</td>
<td>The system shall protect itself from malicious encoding mechanisms.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AC-3</td>
<td>If no identification is provided, the system shall record the security event and notify the operator within a reasonable time</td>
<td>Access Control</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>EN-1</td>
<td>The system’s data and communication shall be encrypted.</td>
<td>Encryption</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PV-4</td>
<td>Users should log out of AMS system or close browser as soon as their activities are done</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AU-2</td>
<td>Audit information must be reviewed routinely.</td>
<td>Auditing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>PV-2</td>
<td>Enforce strong password policies</td>
<td>Privacy</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UA-8</td>
<td>The system shall protect itself from input validation attack by assuring file names are well formed.</td>
<td>Unauthorized Attacks</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SU-3</td>
<td>Only System Administrators are permitted to</td>
<td>Survivability</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>install any software and/or hardware</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---</td>
<td>----------------------------------------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>UA-6</strong></td>
<td>The system shall enforce the use of firewall technology.</td>
<td>Unauthorized Attacks</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EN-3</strong></td>
<td>A secure communication channel shall be used to secure Web data transfer.</td>
<td>Encryption</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td><strong>SU-2</strong></td>
<td>All installation must be approved and reviewed by managers.</td>
<td>Survivability</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td><strong>EN-2</strong></td>
<td>Secure technologies shall be used to provide secure communications channels.</td>
<td>Encryption</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td><strong>UA-7</strong></td>
<td>A form of intrusion detection shall be implemented in the system. If the system detects any corruption of data or messages, then the system shall record the security event; notify the system administrator in a timely manner.</td>
<td>Unauthorized Attacks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
REFERENCES


http://www.sei.cmu.edu/library/abstracts/reports/04sr015.cfm