Towards a Usable-Security Engineering Framework for Enhancing Software Development

by

Yasser M. Hausawi

A thesis submitted to the College of Engineering at Florida Institute of Technology in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Melbourne, Florida
May 2015
We the undersigned committee hereby approve the attached thesis

Towards a Usable-Security Engineering Framework for Enhancing Software Development

by

Yasser M. Hausawi

______________________________________________________________
William H. Allen, Ph.D.
Associate Professor, Computer Sciences & Cybersecurity, College of Engineering
Committee Chair, Thesis Advisor

______________________________________________________________
Phil Bernhard, Ph.D.
Associate Professor, Computer Sciences & Cybersecurity, College of Engineering
Thesis Committee

______________________________________________________________
Liam Mayron, Ph.D.
Assistant Professor, Computer Sciences & Cybersecurity, College of Engineering
Thesis Committee

______________________________________________________________
Muzaffar Shaikh, Ph.D.
Professor, Engineering Systems, College of Engineering
Thesis Committee

______________________________________________________________
Richard Ford, Ph.D.
Professor, Computer Sciences & Cybersecurity, College of Engineering
Department Head
Abstract

Title: Towards a Usable-Security Engineering Framework for Enhancing Software Development

Author: Yasser M. Hausawi

Committee Chair: William H. Allen, Ph.D.

Computer systems are fundamental tools for almost every single process in life. People from all over the globe use computer systems for an unlimited number of purposes. Consequently, a close relationship between people and computer systems exists. Current research investigates how people and such systems to interact with each other in a proper manner. The research work on this matter is being conducted in different directions; one direction is investigating people’s behaviors toward computer systems, such as the cognitive and mental state of humans when interacting with computer systems; another direction studies the computer system’s behaviors toward people, such as studying the systems’ quality attributes and enhancing them during the Software Development Life-Cycle (SDLC) phases; moreover, a third hybrid research area takes a bidirectional approach by examining both people and computer systems. Examples of this hybrid approach are Human-Computer Interaction (HCI), and Human Computer Interaction and Security (HCI-SEC).
Our research project applies the hybrid bidirectional research path of the HCI and the HCI-SEC through investigating two software quality attributes, security and usability, from the human-centered perspectives during three phases of the SDLC, which are requirements, design, and evaluation (testing). The work focuses on assessing, balancing, measuring, and evaluating the two quality attributes and the relationship between them, namely: usable-security.

Towards achieving our goal, we have proposed and designed the Usable-Security Engineering Framework (USEF) for enhancing software development processes when aligning and balancing security and usability is an issue. The framework has three components, each component addresses one of the three previously mentioned phases of the Software Development Life-Cycle (SDLC). The first component is an Assessment Framework for Usable-Security (AFUS) based on decision science during the requirements engineering phase. The second component is a set of usable-security guidelines applied during the design phase. The third component is a usable-security measuring matrix applied during the evaluation phase. The three components are related to one another and work together as one solution that enhances the process of integrating security and usability during software development; however, each component can work independently from the other components.
# Contents

## 1 Introduction

1.1 Overview and Motivation ................................................. 1
   1.1.1 Project Goals .................................................. 2
1.2 State of the Art and Problem Determination ......................... 3
1.3 Thesis Statement ................................................................ 5
1.4 Contributions .................................................................. 8
1.5 Organization .................................................................. 10

## 2 Literature Review

2.1 Background Information .................................................... 13
   2.1.1 Quality Attributes ................................................ 13
   2.1.2 Software Development Life-Cycle (SDLC) ................... 38
   2.1.3 Integrating Security and Usability ............................. 53
   2.1.4 Human-Computer Interaction (HCI) ......................... 57
2.2 Related Work .................................................................. 67
   2.2.1 Human-Computer Interaction and Security (HCI-SEC) .... 67
   2.2.2 Security Assessment .............................................. 89
2.2.3 Usability Assessment ........................................... 90
2.2.4 Usable-Security Assessment ................................. 92
2.2.5 Usable-Security Authentication Design ..................... 97
2.2.6 Hybrid Approaches ............................................. 99

3 Usable-Security Assessment during Requirements Engineering 102
3.1 Introduction ......................................................... 103
3.2 Assessment Framework for Usable-Security (AFUS) ............... 104
  3.2.1 Requirements Filtering and Merging ......................... 105
  3.2.2 Utility Functions ............................................. 112
  3.2.3 Decision Trees .............................................. 116
3.3 Results and Discussion ........................................... 120
3.4 Conclusion ......................................................... 122

4 Usable-Security Guidelines during Software Design 123
4.1 Introduction ......................................................... 124
4.2 Guidelines for Usable-Security ................................. 125
  4.2.1 Selection of Priority Sensitivity ...................... 126
  4.2.2 Requirements Filtering .................................. 128
  4.2.3 Selection of Appropriate Design Tools .............. 128
  4.2.4 Creation of a Linkage Mapping Table .............. 129
  4.2.5 Generation of New Design Tools ..................... 129
  4.2.6 Analysis of the Trade-Off Matrix .................. 130
  4.2.7 Checking the Match between Requirements and Design Tools . 130
  4.2.8 Incorporation of Results and Feedback ............ 131
4.3 An Example: Secure and Usable Authentication .................. 131
4.3.1 Selection of Priority Sensitivity ........................................ 132
4.3.2 Requirements Filtering .................................................. 133
4.3.3 Selection of Appropriate Design Tools .............................. 133
4.3.4 Creation of a Linkage Mapping Table ................................. 133
4.3.5 Generation of New Design Tools ..................................... 135
4.3.6 Analysis of the Trade-Off Matrix ..................................... 136
4.3.7 Checking the Match between Requirements and Design Tools . 137
4.3.8 Incorporation of Results and Feedback ............................. 138
4.4 Conclusion ........................................................................ 138

5 Usable-Security Measurement during Evaluation .................... 139
5.1 Introduction .................................................................. 140
5.2 Usability-Security Measuring and Evaluation ...................... 141
  5.2.1 Usability Evaluation ..................................................... 141
  5.2.2 Security Evaluation ..................................................... 147
5.3 Usable-Security Evaluation Matrix .................................... 154
5.4 Case Study: Authentication Approaches ............................ 157
5.5 Conclusion .................................................................. 162

6 Framework Evaluation: Heuristic and Comparative Experiment 163
6.1 Introduction ............................................................... 164
6.2 Heuristic Evaluation on The Framework ............................ 167
  6.2.1 Overview ............................................................... 167
  6.2.2 Objectives .............................................................. 168
  6.2.3 Heuristic Evaluation Design ....................................... 169
  6.2.4 Heuristic Evaluation Results and Discussion ................. 172

vii
B Applying the AFUS

B.1 Requirements Filtering and Merging . . . . . . . . . . . . . . . . . 255
B.2 Utility Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 260
B.3 Decision Tree . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 262
B.4 Assessment Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 264

C Institutional Review Board - IRB 266

D Determining the Optimal Balance between Security and Usability 276
List of Figures

1.1 The Usable-Security Engineering Framework (USEF) ........................................ 6

2.1 Computer and Information Security Goals ...................................................... 16
2.2 The Seven Security Patterns ......................................................................... 18
2.3 A Sample of Security Patterns ......................................................................... 19
2.4 Information Privacy ............................................................................................ 21
2.5 Privacy Patterns ................................................................................................. 25
2.6 Security and Privacy. .......................................................................................... 27
2.7 Usability Process ................................................................................................ 31
2.8 Usability Properties ............................................................................................ 32
2.9 Interaction of Usability Properties ...................................................................... 33
2.10 Samples of the Selected 51 Usability Techniques ............................................. 34
2.11 Usability Attributes, Properties, and Patterns. .................................................. 37
2.12 A Usability Framework ...................................................................................... 37
2.13 SAFECODE Security Practices ......................................................................... 39
2.14 The seven Touchpoints ...................................................................................... 40
2.15 The Microsoft SDL ............................................................................................ 41
2.16 The ISDF Framework ......................................................................................... 43
2.17 Mapping Usability Techniques to Development Activities . . . . . . . . 46
2.18 The Development Process . . . . . . . . . . . . . . . . . . . . . . . . . 47
2.19 Mapping personas’ supports to requirements activities . . . . . . . . 50
2.20 Usability Engineering and Software Engineering . . . . . . . . . . . . 52
2.21 Human-Centered Design Approach . . . . . . . . . . . . . . . . . . . . 58
2.22 Strategies for Designing Questionnaires . . . . . . . . . . . . . . . . . 61
2.23 Human Participants Research Process . . . . . . . . . . . . . . . . . . 64
2.24 Experiment’s Tasks . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
2.25 Experiment’s Snippet Lengths . . . . . . . . . . . . . . . . . . . . . . 66
2.26 Usable-Security . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
2.27 Pay-Off Matrix for Polite Security Interaction . . . . . . . . . . . . . 72
2.28 The Appropriate and Effective Guidance for Information Security . . . 74
2.29 Sesame . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
2.30 HBSI Evaluation Method . . . . . . . . . . . . . . . . . . . . . . . . . 80
2.31 HBSI Framework . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
2.32 The UTAUT Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
2.33 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
2.34 Objective and Subjective Measures . . . . . . . . . . . . . . . . . . . . . 83
2.35 Usable Security Principles . . . . . . . . . . . . . . . . . . . . . . . . 85
2.36 A Sample Principle . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
2.37 The 21 Usable Security Patterns . . . . . . . . . . . . . . . . . . . . . 87
2.38 A Sample Pattern . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
2.39 The OWASP RRM Overall Risk Severity Subjective Rates . . . . . . . 89
2.40 Evaluating the Design Examples . . . . . . . . . . . . . . . . . . . . . 98
2.41 Usable Security Principles . . . . . . . . . . . . . . . . . . . . . . . . 101
3.1 The Assessment Framework for Usable-Security (AFUS)  
3.2 The Decision Tree for Scenario 1  
3.3 The Decision Tree for Scenario 2  
3.4 The Decision Tree for Scenario 3  
3.5 The ”Before” and ”After” Assessment Results  
4.1 The Proposed Usable-Security Design  
5.1 The Usable-Security Evaluation Matrix  
5.2 Usability and Security Measuring Values  
5.3 Final Usable-Security Evaluation and Categorization Guidance  
5.4 Usable-Security Evaluation Values for All Scenarios  
5.5 Radar Chart for the Usable-Security Evaluation  
6.1 The Two-Dimension Design of the Heuristic Evaluation  
6.2 The CBAA Signing-Up Window  
6.3 The CBAA Logging-In Window  
6.4 The Results of Heuristic Evaluation  
6.5 The Decision Tree for the CBAA  
6.6 Concept of Experiment Operation  
6.7 The System Usability Scale Survey  
6.8 The Design Type of the Comparative Study  
6.9 The Means of the Simple Dependent Variables  
6.10 The Means of the Complex Dependent Variables  
6.11 The Means of Usable-Securitys  
B.1 The Security Requirements
B.2 The Usability Requirements . . . . . . . . . . . . . . . . . . . . . . . 257
B.3 The Initial Usable-Security Requirements . . . . . . . . . . . . . . . 258
B.4 The Merged Usable-Security Requirements . . . . . . . . . . . . . . . 259
B.5 Applying Utility Functions . . . . . . . . . . . . . . . . . . . . . . . . 261
B.6 Applying Decision Tree . . . . . . . . . . . . . . . . . . . . . . . . . . 263
B.7 The Assessment Final Results . . . . . . . . . . . . . . . . . . . . . . 265
D.1 All Possible Values of Usable-Security . . . . . . . . . . . . . . . . . 277
D.2 The Trends of Security and Usability Values . . . . . . . . . . . . . . 278
D.3 The Trade-Off Chart . . . . . . . . . . . . . . . . . . . . . . . . . . . . 279
## List of Tables

2.1 The Results of Assessing the 180 Questions on the Three Attacks . . . . 96  
2.2 Usability and Security During the SDLC . . . . . . . . . . . . . . . . . 100  
3.1 Security and Usability Properties Importance Rating Guidance . . . . 108  
3.2 Initial Usable-Security Importance Rating Guidance . . . . . . . . . . 110  
3.3 The Assessment Results for Scenarios 1, 2, and 3 . . . . . . . . . . . . 122  
4.1 Linkage Mapping Table . . . . . . . . . . . . . . . . . . . . . . . . . . 129  
4.2 Design Tools Trade-Off Matrix . . . . . . . . . . . . . . . . . . . . . . 130  
4.3 Security Requirements Filtering . . . . . . . . . . . . . . . . . . . . . 132  
4.4 Usability Requirements Filtering . . . . . . . . . . . . . . . . . . . . . 134  
4.5 The Linkage Mapping Table for the Security Requirements . . . . . . 136  
4.6 The Linkage Mapping Table for the Usability Requirements . . . . . . 137  
4.7 Design Tools Trade-Off Matrix for the CBAA Demonstration . . . . . . 137  
5.1 The Scenarios of the CBAA . . . . . . . . . . . . . . . . . . . . . . . 159  
5.2 The Scenarios of the TBAA . . . . . . . . . . . . . . . . . . . . . . . . 160  
6.1 The Ranking Terms . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171  
6.2 The Final Results of the Heuristic Evaluation . . . . . . . . . . . . . 172
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3 Security Requirements Filtering and Rating</td>
<td>185</td>
</tr>
<tr>
<td>6.4 Usability Requirements Filtering and Rating</td>
<td>186</td>
</tr>
<tr>
<td>6.5 The Initial Usable-Security Requirements Filtering and Rating</td>
<td>187</td>
</tr>
<tr>
<td>6.6 The Merged Usable-Security Requirements Rating</td>
<td>188</td>
</tr>
<tr>
<td>6.7 Assessing Security and Usability Requirements</td>
<td>188</td>
</tr>
<tr>
<td>6.8 The Ratios of the Security Properties</td>
<td>188</td>
</tr>
<tr>
<td>6.9 The Ratios of the Usability Properties</td>
<td>189</td>
</tr>
<tr>
<td>6.10 The Pointing out Process of the Security Properties</td>
<td>189</td>
</tr>
<tr>
<td>6.11 The Pointing out Process of the Usability Properties</td>
<td>189</td>
</tr>
<tr>
<td>6.12 The Weights of the Security Properties</td>
<td>190</td>
</tr>
<tr>
<td>6.13 The Weights of the Usability Properties</td>
<td>190</td>
</tr>
<tr>
<td>6.14 The Utility Values of the Security Properties</td>
<td>190</td>
</tr>
<tr>
<td>6.15 The Utility Values of the Usability Properties</td>
<td>191</td>
</tr>
<tr>
<td>6.16 The Ratios of the Quality Attributes</td>
<td>191</td>
</tr>
<tr>
<td>6.17 The Weights of the Quality Attributes</td>
<td>191</td>
</tr>
<tr>
<td>6.18 The Utilities of the Quality Attributes</td>
<td>192</td>
</tr>
<tr>
<td>6.19 Assessment for the CBAA</td>
<td>192</td>
</tr>
<tr>
<td>6.20 Security Requirements Filtering</td>
<td>195</td>
</tr>
<tr>
<td>6.21 Usability Requirements Filtering</td>
<td>197</td>
</tr>
<tr>
<td>6.22 The Linkage Mapping Table</td>
<td>199</td>
</tr>
<tr>
<td>6.23 The Linkage Mapping Table for the Usability Requirements</td>
<td>199</td>
</tr>
<tr>
<td>6.24 Design Tools Trade-Off Matrix</td>
<td>200</td>
</tr>
<tr>
<td>6.25 Contribution and Time Duration of the Experiment</td>
<td>211</td>
</tr>
<tr>
<td>6.26 The Results of t-Test on Simple Variables</td>
<td>214</td>
</tr>
<tr>
<td>6.27 The Results of t-Test on Complex Variables</td>
<td>214</td>
</tr>
</tbody>
</table>
6.28 The Results of t-Test on Usable-Security Level ................. 214
6.29 The Results of F-Test on Simple Variables ...................... 215
6.30 The Results of F-Test on Complex Variables ..................... 215
6.31 The Results of F-Test on Usable-Security Level .................. 216
6.32 Overall Pearson’s $r$ Values of Simple Variables ............... 223
6.33 Overall Pearson’s $r$ Values of Complex Variables .............. 224
6.34 The CBAA Pearson’s $r$ Values of Simple Variables ............ 225
6.35 The CBAA Pearson’s $r$ Values of Complex Variables .......... 227
6.36 The TBAA Pearson’s $r$ Values of Simple Variables ............ 228
6.37 The TBAA Pearson’s $r$ Values of Complex Variables .......... 230
6.38 Average Pearson’s $r$ Values of Simple Variables .............. 231
6.39 Average Pearson’s $r$ Values of Complex Variables ............ 231
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFUS</td>
<td>Assessment Framework for Usable-Security</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Attributes</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life-Cycle</td>
</tr>
<tr>
<td>SEC</td>
<td>SECurity</td>
</tr>
<tr>
<td>USA</td>
<td>USAbility</td>
</tr>
<tr>
<td>USA-SEC</td>
<td>USAble-SECurity</td>
</tr>
<tr>
<td>USEF</td>
<td>Usable-Security Engineering Framework</td>
</tr>
</tbody>
</table>
Acknowledgements

First and foremost, I thank our creator, Allah, for his continual great blessings and bestowing mercy on me to complete the parts of this work. His help and support have guided me not only during the course of this dissertation, but since the first moments of my life.

This work has been completed under my name. However, there are many people stand behind every single part of it, without whom this work would not have been possible to complete. Therefore, they deserve special thanks.

My deepest thankfulness and gratitude to my mentor and advisor, Dr. William H. Allen. Without his encouragement, help, and support, I would never have reached to my scientific and academic goals as a researcher and scientist. Dr. Allen is always open to new ideas that are based on solid and robust sciences. Being open minded, Dr. Allen has given me the freedom for self-searching and exploration. As a result of his great and unlimited expertise on managing and controlling academic research projects, and his amazing ability on guiding, advising, and sharpening skills of graduate students, this research work has been successfully achieved, and my academic maturity and capabilities have grown to make me a real self-dependent scientist and researcher. Dr. Allen, you deserve all the meanings of gratefulness, so thank you very much for every single minute you have spent to make this work achievable and completed.
Dr. Liam M. Mayron has remarkable fingerprints on my successfulness as a researcher. His encouragement and enthusiasm have always inspired me to get my hands dirty with research work. Dr. Mayron, thank you for the great help and support.

Dr. Gisela Susanne Bahr has spent much time and great efforts to equip me with the essential skills and tools that I need to develop as a researcher. She is always there to provide thoughtful insight and resources without which this dissertation would have not been completed. Dr. Bahr, I really appreciate your unlimited willingness to help.

The members of my dissertation committee, Dr. Philip J. Bernhard, and Dr. Muzaffar A. Shaikh, have dedicated considerable time and effort to provide me with pieces of advice that indeed enhanced the quality of this dissertation. Dr. Bernhard and Dr. Shaikh, I do thank you for the generosity of time and expertise.

My sponsor, The government of Saudi Arabia (through The Institute of Public Administration in Saudi Arabia IPA), made my dream of pursuing my Ph.D. possible through granting me a scholarship to study in the U.S. They took care of everything from beginning to end (tuition fees, life expenses, medical treatments, trips). Now it is my turn to facilitate my knowledge and expertise to benefit my country.

My warmest tribute, thankfulness, gratitude, and gratefulness to the best person in my life, my giving mother, Ms. Zainab M. Alhausawi, for her unlimited generosity, encouragement, help, and support. She sacrificed a lot to have me continue my education. My mom, your sacrifice has paid-off and your daily calls and prayers have worked.

My best friend, Mr. Omar M. Hawsawi, has been always there to help me with whatever kind of help I may need for my home country during both of my good days and crises. He always encourage and support me. With his being an important part of my life, I really feel safe.
My family and friends, both in my home country or the U.S., I really thank you all for your motivation, inspiration, help, and support, without which, my Ph.D. journey would not have been enjoyable.

Finally, I owe a tremendous debt of gratitude to my wife, Eman A. Mareh, who joined me and experienced every single minute of my long educational journey that has started since 2004. During my journey, she acted all roles (wife, mother, sister, daughter, friend, consultant, advisor, trouble and peacemaker etc.). She took care of our home and kids, and shared each minute of my emotional and financial difficulties. To my dear wife I say, no word can describe my thanks to you. All I can say that from now on it is my turn to pay you and our kids back for the missed days.
Declaration

I declare that the work in this thesis is solely my own except where attributed and cited to another author. Most of the material in this thesis has been previously published by the author. For a complete list of contributions, please refer to Appendix A at the end of this thesis.
This dissertation is dedicated to:

The memory of my great father, Mr. Mohammed Hausawi, who past away when I was four years old; his dream will always be an inspiration.

My affectionate mother, Ms. Zainab Alhausawi, who always trusts and believes in my ability to achieve my dreams; her sacrifices will always be the foundation of my successfulness.

My lovely wife, Mrs. Eman Mareh, who always keeps motivating me to our advancements; her love and support will always be my fuel to work hard.

My source of happiness, my sweet children: Tafaul, Mohammed, and Farah; their smiles always keep me happy and optimistic.
Chapter 1

Introduction

1.1 Overview and Motivation

With the revolution of digital technology, computing systems have become the most important tools that ease the intricate nature of science and engineering. Computing systems are targeted to make use of science and engineering theories and methods applicable in real life.

The computing systems of science and engineering mainly interact with humans. However, interaction with the human is not an easy task, especially human-machine interactions; therefore, in order for such interaction to proceed in a proper way, a considerable understanding of the human nature of interaction and the way that such interaction is assessed, designed, and evaluated on the SDLC must be achieved. As the better we understand, assess, design and evaluate the human-computer interaction, a better synergetic relationship can be developed. This leads us to one important term that plays a major role in better human-computer interaction, which is usability. Although usability is a very important goal to achieve for Human-Computer Interaction, it seems
to be in conflict with another more important goal of computer systems, which is security [30].

Unfortunately, both security and usability are considered as nonfunctional quality attributes [21]. Therefore, they are usually not considered at the early stages of developing software products; rather, their techniques, mechanisms, and best practices are applied as add-ons to user interfaces after the product’s development is almost completed [7]. Consequently, their requirements lack proper assessment, which is an important step in predicting the levels of the two attributes once their requirements are applied during the later phases of the SDLC. More importantly, the effect of integrating the requirements of both of the attributes is not assessed. This lack of assessing the requirements of security and usability attributes along with their interaction (usable-security), creates a gap that negatively affects applying both of the attributes properly during the later phases of the SDLC. As a result, although there have been numerous approaches, methods, and techniques to design and evaluate software products that achieve a synergetic interaction between security and usability, we have not yet reached such goals satisfactorily. Our research work focuses on three of the SDLC phases: requirements, design, and evaluation (testing). Our dissertation works towards improving the process of applying better usable-security on the SDLC through requirements assessment, HCI-SEC design approach, and evaluation.

1.1.1 Project Goals

The overall objective of our research project is to develop a framework that helps enhancing the alignment of security and usability, and achieving better usable-security for software systems development. We focus on assessing and balancing security and
usability to provide a synergetic relationship between the two attributes that leads to good measurable usable-security. Our research aims for three goals:

1. Enable developers to establish the correct balance between security and usability by applying decision science techniques to the requirements engineering phase of the software development life-cycle.

2. Provide guidelines to assist developers in selecting the best set of design tools available so they can employ a usable-security approach during the design phase.

3. Design and demonstrate a usable-security evaluation methodology that can be used during the evaluation (testing) phase to assess a developed products’ compliance with the usable-security requirements and design specifications that were created for that product.

1.2 State of the Art and Problem Determination

Integrating usability and security aiming for acceptable balance is a challenging issue [26]. Although there are a considerable number of studies, research papers, and experiments trying to address this issue through providing new design methodologies and proposing several approaches, there have not been perfect solutions so far. Hence, there is still enough room for further research to work within the scope of the intricate nature of seeking optimal ways to integrate usability and security in a proper and effective manner [100, 91]; thus, many ideas can be proposed as extensions to the already existing proposed solutions. In addition, new ideas could be proposed to compete with the currently existing practices. The battle to overcome the challenge of balancing usability and security integration is ongoing and it does not appear that it
will be resolved soon [91]. Once the security mechanisms become sufficiently usable for users, the security practitioners along with usability specialists can celebrate the end of unintentional conflict with genuine users [36]; the security practitioners can then limit their focus to beating adversaries. Lack of applying acceptable usable-security during the phases of the SDLC forces security experts to put both genuine users and adversaries in one basket. They must try to protect computer systems against both by only applying security engineering processes to the SDLC phases with respect to the fact that security is mostly considered an add-on quality attribute. On the other hand, lack of applying acceptable usable-security during the phases of the SDLC forces usability experts to ignore security and limits their work toward making software products usable enough to all users without distinguishing between genuine users and adversaries.

However, despite the work on the two quality attributes and the integration between them - namely, security, usability, and usable-security - there is an obsessive need to utilize the outcomes of such research work into the SDLC. The currently published research studies have focused on three directions:

1. Integrating a particular quality attribute (i.e. security or usability) into the SDLC.

2. Integration between the two attributes starting from the design phase of the SDLC.

3. Measuring the two attributes separately with neither consideration to interrelating them (usable-security) nor the possible side effects of the conflict of interest between the two attributes.

Consequently, achieving acceptable usable-security on software products is still problematic for four reasons.

First, the nature of the relationship between security and usability is not clear enough, as there are three different viewpoints about such a relation. One view looks
at security and usability as conflicting attributes [1, 67, 72, 102, 105]; another view looks at the two attributes as synergetic and, therefore, claims they must be treated equally [12, 29]; and a third view prioritizes security over usability by making security the primary attribute that need to be usable enough [26, 38, 89].

Second, both security and usability specialists believe that in order to achieve acceptable usable-security, both attributes should be considered at the earliest stage of the development process; however, requirements engineering is the first phase of the SDLC. Security and usability requirements are derived at that stage, but there is no assessment tool currently available to predict the impact of both of the attributes’ requirements and the impact of their interaction (usable-security). Moreover, most of the research work is done in favor of the design phase of the SDLC.

Third, as a direct side effect from the lack of work on usable-security at the requirements engineering phase, the research work at the design phase experiences weaknesses because the work is not based on the previous phase (requirements) of the SDLC [7].

Fourth, both security and usability are considered subjective quality attributes; therefore, they are evaluated subjectively. As a consequence, they lack reliable objective measures and metrics to evaluate the two attributes and their interaction.

1.3 Thesis Statement

This research investigates three phases of the SDLC, requirements, design, and evaluation (testing) through proposing a framework that consists of set of sequenced techniques and methods that will help software engineers to achieve better usable-security as depicted in Figure 1.1.
Figure 1.1: The Usable-Security Engineering Framework (USEF). It consists of three components (Usable-Security Assessment - AFUS, Usable-Security Guidelines, and Usable-Security Measuring Matrix). Each component focuses on one of the three phases (Requirements Engineering, Design, and Evaluation / Testing) of the Software Engineering Development Life-Cycle (SDLC) in order to enhance aligning security and usability for better usable-security.
The framework works towards addressing the four problems (described in Section 1.2) that hinder improving the processes during the SDLC for better usable-security. The framework uses well-known tools, approaches, and principles from different disciplines, such as decision science and Human-Computer Interaction and Security (HCI-SEC), along with the discipline of computer science in order to improve the process toward better usable-security.

During the requirements engineering phase, our mechanism proposes an assessment technique in order to investigate usability and security along with the interaction between them though predicting the levels of security, usability, and usable-security based on the requirements. The assessment benefits from utility functions and the decision trees, well-known tools of the decision science field. We called the technique an Assessment Framework for Usable-Security (AFUS).

During the design phase, our framework proposes an initial design method that provides a strong relationship between the requirements and the design tools such as design principles, patterns, best practices, activities, and techniques. The primary goal of this method is to ensure that the design phase is capable of meeting the requirements. Moreover, it provides feedback to the requirements assessment (the AFUS) and provides a mapping table that supports the later design phase processes. The method benefits from the principles of the Human-Computer Interaction and Security, and is based on the AFUS results from the requirements engineering phase.

During the evaluation phase, our framework proposes an evaluation method in order to provide feedback about the level of the usable-security in the requirements and design phases after the systems are developed.

Our research does not propose solutions during the construction phase because this phase includes convenient tools that, if applied properly, would meet the goals of
its preceding phases. Thus, our usable-security framework provides a comprehensive process that targets the usable-security related key phases starting from the initial phase of the SDLC. Moreover, unlike other usable-security solutions currently available, our framework provides a consecutive set of usable-security processes that feed into each other as a sub-cycle within the SDLC.

To better illustrate the advantages of our framework, we have developed a demonstration sampler based on a proposed approach called the Choice-Based Authentication Approach (CBAA), and we incorporated our framework through the requirements phase and the beginning of the design phase.

1.4 Contributions

The key contributions of this research work are:

- **Development of an Assessment Framework for Usable-Security based on Decision Science AFUS**: This work is not to produce an objective measure for comparing two products, but rather to generate a metric that developers can use to gauge the balance between the attributes. We assume that the developers of a product are aware of the balance between security and usability that is appropriate for their product, thus the proposed technique is intended to assist in reaching that desired balance. As changes to the requirements are made, reassessment using AFUS can indicate if the product has shifted to a greater emphasis on one attribute at the expense of the others, or if all attributes have moved towards the developer’s preferred equilibrium. This work is presented in Chapter 3.
• Development of Guidelines for Resolving Trade-Offs Between Security and Usability in Software Design: This work provides systematic guidance for linking the usability, security, and usable-security requirements to appropriate design tools (i.e. principles, practices, patterns, and techniques) that can be used during the architectural design stage. The assessment of the AFUS is used as key evaluation method that supports the linking process. Chapter 4 presents this contribution in details.

• Development of a Usable-Security Measuring Matrix: This work proposes a summative usable-security evaluation matrix based on the usable-security assessment models and frameworks. It serves as a final checkpoint for applying the AFUS and the guidelines in order to validate the resolved trade-offs between security and usability. The evaluation matrix is proposed in Chapter 5.

• Conducting a Heuristic Evaluation Survey on the framework: The survey on the framework is conducted in order to evaluate each of the framework’s components individually. Moreover, it evaluates the integration between the components when they work together as a single usable-security processing framework. The survey is presented in Section 6.2.

• Design and implementation of a Choice-Based Authentication Approach CBAA: This contribution proposes an authentication system that addresses usability, universal access, and security issues based on two concepts. The first concept is to allow end users to select their authentication method based on their
preferences in order to provide better usability and universal access. The second concept is to increase the difficulty for adversaries by displaying all of the possible authentication methods at one time, increasing the complexity of guessing the user’s chosen authentication approach. The CBAA is used as a case-study to illustrate the application of the developed components, namely the AFUS, the guidelines, and the evaluation matrix. This is presented in Section 6.3.

- **Conducting a Comparative Study between Choice-Based and Standard-Based Authentication and Traditional-Based Authentication Approach on Usable-Security**: This contribution investigates whether the proposed approach raises security and provides usability and universal access without increasing authentication complexity, through conducting a comparative experimental research that compares between the proposed CBAA and the Traditional-Based Authentication Approach - alphanumeric password (TBAA). This study is presented in Section 6.4.

1.5 Organization

The remaining of this dissertation is structured as follows:

- **Chapter 2**: literature review on the software quality attributes under investigation: security, usability, and usable-security. In particular, the chapter presents background information on security, usability, and usable-security. In addition,
it presents related work on integrating security and usability together in order to form usable-security.

- **Chapter 3**: an Assessment Framework for Usable-Security based on Decision Science *AFUS* during the requirements engineering phase of the Software Development Life-Cycle (SDLC).

- **Chapter 4**: a Guidelines for Resolving Trade-Offs Between Security and Usability in Software Design during the design phase of the Software Development Life-Cycle (SDLC).

- **Chapter 5**: a Usable-Security Measuring Matrix for evaluating developed software products during the evaluation (testing) phase of the Software Development Life-Cycle (SDLC).

- **Chapter 6**: heuristic evaluation survey and comparative experiment for validating the usable-security framework.

- **Chapter 7**: conclusion and future work.
Chapter 2

Literature Review

Preface

This chapter presents detailed review on usable-security literature that is used as background knowledge and related work for conducting the research work throughout the remainder of this dissertation.

Background information of relevant topics from the fields of software quality attributes, software development life-cycle, and human-computer interaction are presented in Section 2.1. The software quality attributes field in this dissertation focuses on the non-functional attributes: security, privacy, and usability (Section 2.1.1). The software development life-cycle focuses on integrating the attributes separately into the development life-cycle (Section 3.3). In addition, Section 2.1.3 presents background knowledge on integrating security and usability together during the development process. Finally, Section 2.1.4 presents background information about the human-computer interaction, which is a field that is concerned with the design, implementation, and evaluation of interactive computing systems for human use.
Surveying and reviewing the related research work on usable-security is presented in Section 2.2. Previous approaches, solutions, methods, and mechanisms are presented, as they are used to initiate the research work on usable-security.

In addition, Section 2.2.6 presents related contributions on usable-security that we have participated in and published through conferences.

2.1 Background Information

2.1.1 Quality Attributes

Since the main idea of this report is to integrate usability and security together in order to develop Usable Security mechanisms aiming to secure computer systems in a proper way, it is worthwhile to provide some background information about software quality attributes in general. In addition, the three strongly related attributes of "Privacy, Usability, and Security" will be discussed in particular. Note that privacy is not considered as a quality attribute and it is included under security, as privacy is equated to confidentiality [45, 104].

There are two types of privacy: 1) physical privacy that includes any private belongings that can be accessed physically, and 2) electronic privacy that includes any private belongings that can be accessed digitally. Security also has two ways to be ensured: 1) physical security, and 2) electronic security. In same way, usability is also divided into two parts: 1) physical usability, and 2) electronic usability. As computer systems deal with the electronic aspect of technology, the scope of this report is limited to electronic privacy, security, and usability.
Security:

The term *Security* is defined in many ways. Garfinke and Spafford define a computer as secure, "if you can depend on it and its software to behave as you expect it to" [43]. Pfleeger and Pfleeger define computer security as, "preventing the weaknesses from being exploited and understanding preventive measures make the most sense" [80].

Essentially, system security is a set of methods and techniques that work together to generate what is called *Security Mechanisms*. The security mechanisms are used to prevent weaknesses of computer systems from being exploited by applying three security properties: 1) *confidentiality*, 2) *integrity*, and 3) *availability* [80]. OWASP Risk Rating Methodology (OWASP RRM) adds one more property *accountability* and divides it into two sub-properties: non-repudiation and auditing [78]. Security mechanisms are spread throughout computer systems because most of computer systems deal with sensitive and private information in various ways. The goal of any security mechanism is to fulfill and achieve the three security properties.

*Security* is related to software in many ways. *Software Security* and *Security Software* are two different joint terms where security and software work together to ensure security; however, the two terms are not the same. *Software Security* is a process conducted during the software development lifecycle to apply security practices in order to produce secure software products [75]. Conversely, *Security Software* is a software product developed to protect information or resources [75].

Security Properties:

Confidentiality, integrity, availability, and accountability must be applied in order to provide secure computer systems. The following is a brief definition for the four security goals of computer systems security [80, 78]:
• **Confidentiality:** The ability for the security mechanisms to grant access only to authorized users. Confidentiality can be measured objectively by measuring the rates of false acceptance and false rejection.

• **Integrity:** The ability for the security mechanisms to allow only authorized users to only perform tasks in a proper way, and protect the data on computer systems from unauthorized alteration. Integrity can be measured objectively by hashing match (MD5, AH1). Integrity is a crucial goal because information reliability relies on integrity.

• **Availability:** The ability of the security mechanisms to ensure that computer systems are available for the authorized customers at any time. Security mechanisms must protect computer systems against attacks, such as denial of service (DOS), as some of these attacks happen accidentally via genuine users.

• **Accountability:** The ability of the security mechanisms to monitor and analyze users’ logs to discover and then prevent any violation incident (auditing), and trace-back all users activities to prevent any later denial of committed activities (non-repudiation) [7].

When the above four goals are applied to computer and information systems properly, the system’s security is met. Figure 2.1 explains the interaction between the goals to achieve security.

**Security Patterns:**

The idea of using patterns in design was first introduced in 1975 by the construction architect Christopher Alexander in his book *The Oregon Experiment* [3, 6, 48]. Two years after he introduced the idea of using patterns, Alexander worked with other
researchers to publish the first book on pattern language called *A Pattern Language* [4] that describes a set of 253 patterns used for architectural building and planning. In *A Pattern Language*, Alexander et al. define a pattern as a document that, "describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it twice the same" [4], and cited in [6]. Alexander and his colleagues examined buildings over the globe to develop a pattern language [48]. In 1979, Alexander followed his work of [4] with a more detailed book called *The Timeless Way of Building* [5]. The book described design techniques of using the patterns to decrease building time [6].

There are various different definitions for a pattern other than Alexander et al.’s definition. For instance, Coplien defined a pattern as, "a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to solve themselves" [56]. Alkussayer [6] defined a pattern as a term that "describes a time-tested generic solution
to a recurring problem within a specific context” [6].

According to Heyman et al. [56], there are three general advantages of using patterns. The advantages are:

1. Solutions that patterns provide are tested over the time and proven to properly address particular problems.

2. The strengths and weaknesses of the patterns are well-known before using the patterns.

3. Patterns provide common terms known by stakeholders with different background knowledge.

After Alexander introduced and used patterns in construction design and architecture, using patterns has become widely used by many engineering and science disciplines. Computer scientists, and software engineers and architects adopted using patterns in software designing and programming in 1992 when the first programming pattern language was published by Raiph Johnson [63]. He developed 10 patterns for designing graphical editors [48], and Johnson’s work was followed by the work of Gamma et al., known as the Gang-of-Four (GoF) [42]. Gamma et al. described software patterns as accurate solutions to specific object-oriented software design problems [6]. Since then, the use of patterns in software development has become increasingly reliable and popular. Consequently, Hanmer [50] has published the largest set of general software patterns (63) in his book Patterns for Fault Tolerant Software [48].

As a result of the popularity and reliability of using patterns in software development, security practitioners adopted using the idea of patterns in their work.
Schumacher et al. [94] defined a security pattern as a description of, “a particular recurring design problem that arises in a specific design context, and presents a well-proven generic solution for it. The solution consists of a set of integrating roles that can be arranged to form multiple concrete design structures, as well as a process for creating any particular structure” [94].

Yoder and Barcalow, software security fellows, developed their first set of security patterns in 1997 by [108], this is cited in [6, 48, 56]. Yoder and Barcalow devised seven security patterns for application development. The patterns are listed in Figure 2.2.

In 2006, Schumacher et al. [94] introduces a set of 46 enterprise-oriented security patterns used for fusing security with systems engineering to address enterprise-wide issues [6].

In 2012, Hafiz et al. [48] described a mechanism of growing security pattern language through cataloging, classifying, and relating the patterns. The authors present a total of 96 security patterns for software development usage. Figure 2.3 gives a sample of the 96 security patterns described by Hafiz et al.

There are several ways for documenting security patterns. The Gang-of-Four [42] uses a template that consists of the following characteristics:

- **Name:** states the name of the pattern along with a brief summary.
- **Problem:** describes the problem that the pattern is intended to address.
Figure 2.3: A Sample of Security Patterns by Hafiz et al. [48]

- **Solution:** provides the principle that the pattern relies on to address the problem.

- **Consequences:** lists the advantages of using the patterns.

- **Variations:** describes the variants or specialization of a pattern.

- **Known uses:** provides valid examples where the pattern is successfully used.

The pattern characteristics template of the Gang-of-Four has a wide range of use; however, it does not contain a complete set of pattern characteristics. Alquassayer [6] described a de facto pattern characteristics template that is used by Schumacher and his co-researchers in [94]. The template contains the following characteristics [94]:

- **Name:** states the name of the pattern along with a brief summary.

- **Also Known As:** other names of the patterns

- **Example:** an example from the real-world where a problem happens and the need for the patterns to solve such a problem.

- **Context:** describes the situations where the pattern can be used.
- **Problem:** describes the problem that the pattern is intended to address.

- **Solution:** provides the principle that the pattern relies on to address the problem.

- **Structure:** depicts the structure of the pattern.

- **Dynamics:** provides typical pattern run-time behavior’s scenarios.

- **Implementation:** provides guidance for implementing the pattern.

- **Example Resolved:** discusses any important aspects for resolving the example.

- **Variations:** describes the variants or specialization of a pattern.

- **Known Uses:** provides valid examples where the pattern is successfully used.

- **Consequences:** the advantages of using the pattern.

- **See Also:** provides further resources to other patterns addressing the same problems.

In the software development field, security patterns are commonly modeled using the Unified Modeling Language (UML) [6, 56], perhaps because software designers and architects are most likely to adopt using security patterns during software development; however, there are many useful security patterns that cannot be modeled using UML diagrams [56].

**Privacy:**

*Privacy* is an essential human right that motivates security researchers to create methods and techniques to protect private information. Since the revolution of information technology, electronic privacy has been highly related to information technology.
Privacy has a set of five attributes: *grant access, own, alter, publish, and pass*. The attributes are shown on Figure 2.4 and described as the following:

1. **Grant Access**: the right to give privileges for accessing private information.
2. **Own**: the right to own private information.
3. **Alter**: the right to modify or update private information.
4. **Publish**: the right to publish private information to the public.
5. **Pass**: the right to pass private information to a particular party.

Caloyannides [19] wrote an article about the relationship between privacy and information technology. He stated that the information technology revolution made personal information collection, storage, remote retrieval, and mining easy. Governmental and commercial sectors were not able to use such technology to violate privacy, neither through monitoring and controlling nor through directed marketing, until the tragedy of September 11, 2001. According to Caloyannides, "privacy is the right of individuals to control the collection and use of information about
themselves” [19]. The author also quoted Lius Brandeis’, a Justice of the U.S Supreme Court, definition of privacy is ” the right to be left alone.” [19].

After the tragedy of 9/11, the U.S government shifted the traded privacy rights for increased national security. As a result of this shift, the restrictions of the Federal Privacy Protection have been lifted from government agencies; previously, government agencies could only store relevant and necessary personal information. After the shift, the restrictions of the Federal Privacy Protection have been limited to protect information in specific aspects of individual privacy only.

Moreover, the author stated that privacy has different meanings and considerations around the world. Western Europe countries classify privacy as a human right, and therefore it is a high priority demand. However, the U.K. and many other countries protect their citizens’ privacy from other countries, but not from internal surveillance. China has a privacy protection law, but it is not strictly enforced. In Cuba and Iraq, citizens’ privacy protection is violated and disregarded. According to the author, a fundamental conflict exists between the philosophies of privacy from different positions’ viewpoints. Law enforcement agencies and secret police officials use private information to protect national security. The commercial sector uses private information to know customers’ preferences and needs. Civil libertarians are against using individuals’ privacy by the government and the commercial sector as they are not as trusted as they are supposed to be. Based on the article, there are only two ways to decrease privacy crimes: either by more jails via politicians and caps, or by teaching children the right behaviors and techniques via families. Finally, the author listed some technology that violate privacy such as Global Positioning Systems (GPSs); cameras at key locations, computer and network forensics, and DNA profiling [19].
Moreover, York [109] wrote an article about privacy. He discussed cases where privacy rights are violated by technology, such as when cameras record patients, ATM machine users, traffic lights, and waiting rooms. Those cases are examples of violating privacy rights without an individual’s permission. Furthermore, he listed some recent technology, such as tiny video cameras hidden in devices, mobile phones, door bells, thermostats, clocks, and rear view mirrors in cars, which can be used to gain personal information without prior permission. According to the author, the impact of developing these devices can be against the Fourth Amendment of the U.S. Constitution, that states: "the right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures" [109].

Cybenko [28] wrote about privacy. He stated that people’s expectations of privacy have to change with the current trend of technology development. According to Cybenko, privacy policies should be enforced and applied according to the current technology. The true technology from the privacy viewpoint is the one that gives individuals the right to control their personal information. However, to get such true technology, there is a question to be asked: what degree of privacy should individuals allow in order to use that technology? To answer the previous question, some decided to separate technology from policy and work on each individually. Cybenko stated that such decision is proven to be infeasible, and it is rather better to work on both privacy technology and privacy policy together [28].

**Privacy Patterns:**

Current research has not thoroughly investigated privacy patterns, perhaps because privacy is considered to be a part of security. Additionally, the perception exists that using security patterns is sufficient to take care of privacy. Consequently, limited work
has been done on privacy patterns. Most of the available privacy patterns are suitable for networking and transmission area.

Hafiz [47] wrote a paper on privacy patterns for design purposes. He used messaging systems, dataset release, and location tracking systems to illustrate his patterns. The author named nine privacy patterns all from transmission viewpoint, but the article only described four of them. The four described privacy patterns are displayed on Figure 2.5 and described as the following:

- **Anonymity Set**: mixing data from different sources to hide the link between particular data and its source.

- **Morphed Representation**: changing data representation while passing through a middle transmission point to hide the link between incoming and outgoing data (hiding incoming/outgoing data mining).

- **Hidden Meta-data**: hiding the header information about sources by not linking it to its sensitive data contents.

- **Layered Encryption**: using multi-layer packet encryption to control the data that can be decrypted only for routing purposes.

For the above four privacy patterns, the author used the following 10 characteristic pattern template:

- **Intent**: defines a pattern.

- **Known As**: gives other terms of the pattern.

- **Motivation**: provides examples to illustrate cases.
Figure 2.5: Privacy Patterns by Hafiz [47]
• **Context:** describes what the pattern is supposed to address.

• **Problem:** raises the problem(s) that the pattern is having.

• **Forces:** lists situations and things to be considered when dealing with the pattern.

• **Solution:** proposes solutions to the faced problem(s) and forces.

• **Design Issues:** considers design concerns.

• **Consequences:** presents the benefits and liabilities.

• **Known Uses:** gives sample applications use the pattern.

Moreover, Hafiz [47] briefly defined the remaining privacy patterns:

• **Cover Traffic:** keeping track of nodes in a dummy traffic flow to preserve anonymity and also accountability as well.

• **Batched Routing:** collecting data traffic to a certain threshold for better anonymity results.

• **Delayed Routing:** randomizing the delays in data traffic to prevent traffic time fingerprinting.

• **Constant Length Padding:** making all packets the same size to prevent packet-size fingerprinting.

• **Constant Link Padding:** Using all outgoing nodes evenly to distribute data traffic to prevent traffic link fingerprinting.

A visible drawback may occur when these privacy patterns integrate with security and usability, as the described patterns might violate security and usability properties.
The Relationship between Privacy and Security:

The relationship between privacy and security has been examined from various points of view. Some researchers believe that privacy and security are identical; some others believe that they are completely different, a third group believe that privacy is part of security as depicted in Figure 2.6; while a fourth group looks at the relationship between privacy and security from a different viewpoint as explained by Herold [55] on the following paragraph.

Herold [55] wrote about the differences between security and privacy from the industry viewpoint. According to Herold, privacy is set of legislations that determine the right to own information and permit access, update, publish, and pass on such information. Security is set of tactical strategies, methods, and techniques used to make information unreadable, inaccessible, unaltered, and denied for unauthorized parties through ensuring confidentiality and safety. Herold stated that security professionals are divided into two groups in the way they understand the relationship between the two terms, but neither are true. On one hand, there is a group believes that security and privacy are the same. On the other hand, others believe that the two terms are completely opposed one another. According to Herold, many organizations have two
separate departments: one for privacy considering legal issues, and another for security considering technical issues. Consequently, the two departments do not communicate and each work separately. Herald claimed that such organizational structure is not correct; instead, the two departments should work together and coordinate with one another. The security department’s personnel must be involved in privacy issues. In the same way, the privacy department’s personnel must be involved in security issues. As the relation between privacy and security is consolidated, security is implemented to ensure privacy [55].

Caloyannides [20] wrote another article about the reality of the relationship between privacy and security, and whether privacy constrains security or not. The author started his article by claiming that online monitoring cannot prevent terrorism because it cannot monitor overseas planning and coordination to commit a crime. Therefore, it is ineffective. Moreover, people who work at sensitive positions, like administrators, might use monitored information of individuals inappropriate ways. A third matter is that it is difficult to differentiate between acceptable and unacceptable (abusive) use of privacy. According to Caloyannides, there are two ways to deal with privacy: the first is through law enforcement, while the second is through intelligence. Law enforcement is not successful in the long term because it is proven that attacks have been doubled when such enforcement was used and favored over intelligence after 9/11. Rather, the author proposed using enhanced intelligence forces that are professional, effective, and functional in place of using law enforcement [20].

Usability:

*Usability* is defined by the International Standard Organization (ISO) as the range in which a product can be operated by legitimate users to satisfactorily perform specific
tasks in an effective, efficient, and specified way [59]. Other researchers who are involved in usability add accuracy, memorability, learnability and users knowledge and skills [67]. Garfinkel defines usability as “the measure of the quality of user’s experience when interacting with a product or system whether a web site, a software application, mobile technology, or any user-operated device” [43].

As a result, the usability community ends up with two levels of usability factors. Primary usability factors that are defined by the ISO (effectiveness, efficiency, and satisfaction) [59]. Secondary usability factors are: accuracy, memorability, learnability and users’ knowledge and skills. These secondary factors might be points of intersection with other terms such as security or scalability. The scope of this report focuses on the primary factors that are defined by the ISO; however, some of the secondary factors will be stated or indicated through cited works by other researchers.

**Usability Engineering and its Process:**

Rosson and Carroll [88] define the term *Usability Engineering* as: a set of concepts and techniques that are used to plan, achieve, and verify usability goals and objectives for systems. Hix and Hartson [57] define usability engineering as, “a process through which usability characteristics are specified, quantitatively and early in the development process.”

Good et al. [44] specifies a set of five steps of the usability engineering process as the following: 1) defining measurable usability objectives; 2) setting planned levels to achieve the defined objectives; 3) analyzing the impact of design solutions; 4) incorporating user-derived feedback during and after design stage; and 5) iterating between the design and evaluation until the defined usability objectives are achieved.
Ferre et al. [35] specifies three major phases for usability process, where each phase has a set of activities. Figure 2.7 shows the usability processes as developed by Ferre et al., and the major phases along with their activities are as follows:

1. **Usability Analysis Phase:**
   - User analysis.
   - Task analysis.
   - Usability benchmarks.

2. **Usability Design Phase:**
   - Conceptual design.
   - Visual design.
   - Prototyping.

3. **Usability Evaluation:**
   - Usability testing.
   - Thinking loud.
   - Heuristic evaluation.

**Usability Properties:**

Figure 2.8 shows the usability properties, and the following is a brief definition for the seven primary and secondary *usability properties* [59, 67]:

- **Effectiveness:** the user’s ability to reach his or her goal of operating a targeted computer system. Effectiveness can be measured objectively by the number of
successes of performing tasks on a particular computer system, and reaching the goal of performing such tasks.

- **Efficiency**: the user’s ability to perform a specific task and complete it within an acceptable amount of time. Efficiency can be measured in an objectively quantifiable way by the amount of time consumed to complete a specific task. The acceptable amount of time can be counted through calculating the accumulative average of the past tasks.

- **Satisfaction**: the factor of the user’s happiness of working on a particular computer system. Satisfaction can be evaluated using subjectively qualitative way by the degree of the user’s happiness.

- **Learnability** [35]: the user’s ability to learn how to operate a product. Learnability can be measured subjectively by the number of correct steps during the first time of performing a particular task on the product.

- **Memorability**: the user’s ability to remember how a product is operated and remember the required information to operate the product. Memorability can be measured objectively by counting the number of memorable steps when a user uses a computer system again.
Figure 2.8: Usability Properties

- **Accuracy**: the user's ability to operate a product and get accurate results. Accuracy can be measured objectively by matching previously expected results against outcome results.

- **Error rate** [35]: the number of errors that the user makes while performing a particular task. It is worthwhile to state that error rate is used to measure accuracy, and cannot be considered as an independent usability property.

If users are able to perform a specific task in a particular computer system, and achieve their goal of using such a system within an acceptable amount of time and with a high degree of happiness, the computer system is usable. Figure 2.9 shows the interaction of usability properties in order to achieve system’s usability.

Foltz et al. [41] researched usability engineering. They recalled seven basic design and evaluation criteria of software ergonomics (software usability) and work analysis reported on ISO 9241 "Ergonomics of Human-System Interaction” in 2006. The criteria are: 1) suitability for the task, 2) self-description, 3) controllability, 4) conformity with user expectation, 5) error tolerance, 6) suitability for individualization, and 7) suitability for learning.
According to the authors [41], those criteria, when properly met, can address the two types of HCI confronting usability problems: 1) task-related; and 2) interaction related problems. The authors stated that in order to ensure addressing task-related problems, the design and evaluation criteria must be met through considering task or work analysis. Task or work analysis is divided into three models: normative, descriptive, and formative. The normative model is the best once achieved because it prescribes the optimal behavior of working systems, but it is not achievable due to difficulties when used in systems. The descriptive model investigates the actual behaviors that the users exhibit when working on systems. This model also face difficulties when used in systems. The formative model is a requirements engineering stage model, focusing on both technical and non-technical (organizational) requirements. In other words, the formative model focuses on contextual design and cognitive work analysis; therefore, it overcomes difficulties that the first two models face when used in systems.
Usability Patterns and Techniques:

Folmer and Bosch [39] define the term *Usability Pattern* as, ”a technique or mechanism that can be applied to the design of the architecture of a software system in order to address a need identified by usability property at the requirements stage.”

Ferre [33] surveyed the usability engineering literature to determine usability techniques. A number of 82 techniques have been collected, and among those 82 techniques, only 51 usability techniques are selected that can be included into the software development process. A sample of the selected techniques is presented on Figure 2.10. The selection criteria are:

- Alienation to software engineering.
- General acceptability in the HCI.
- Cost of application.

The selected set of 51 usability techniques are integrated into the software development process as shown in the next section.
In [34] Ferre et al. extended the work of [33], by collecting a set of 95 HCI techniques that consider usability from the HCI literature. The authors characterized the techniques based on the following six criteria:

- User participation (based on user-centered approach).
- Training needs (amount of training needed to apply a technique).
- General applicability (applicability to software development).
- Proximity to software engineering (similarity of a technique’s principles to software principles).
- Usability improvement/effort ratio (role in overall usability enhancement).
- Representativeness (commonly applied to HCI field).

Based on the above criteria, the authors evaluated each of the 95 HCI techniques and gave each technique a total rating as the following:

- Very useful (high in usability improvement/effort ratio, no need for much training, medium to high applicability, and either very close to software engineering or commonly applied in HCI field).
- Useful (medium in usability improvement/effort ratio, medium to high applicability or user participation encouragement).
- Not very useful (none of the above).

Based on the above total rating, a number of 35 techniques have been characterized as either very useful or useful, and selected to be used in a framework for integrating
usability practices into the software development process as we will see in the next section.

Folmer and Bosch [39] have come up with a framework that bridges usability and software architecture. The authors claim that one major reason for not implementing usability in architecture is that several of the usability patterns are architecture sensitive, and consequently they are prohibitively expensive to implement. Another reason for not implementing usability is that most usability issues are only detected late during the summative evaluation of software products, which make it very difficult to modify the system’s architecture due the sensitivity of applying the usability patterns to the already designed architecture. The researchers developed a framework that has a set of usability patterns to be used during software architecture design. The patterns close the gap between usability attributes and usability properties (requirements), as the patterns are specific ways used to link the attributes to their related properties. Figure 2.11 presents Folmer and Bosch’s attributes, properties, and patterns, while Figure 2.12 depicts a sample of their framework.
Figure 2.11: Folmer and Bosch’s Usability Attributes, Properties, and Patterns [39] (Adapted)

Figure 2.12: Folmer and Bosch’s Usability Framework [39] (Adapted)
2.1.2 Software Development Life-Cycle (SDLC)

Applying quality attributes to software is crucial to organizations and their software developers. Quality attributes in general, and security and usability in particular, are currently among the top priority requirements to be applied on most of recently developed software projects. The following sub-sections present the state of integrating security and usability to software through the software development life-cycle.

Integrating Security into Software Development Life-Cycle:

As security became one fundamental quality attribute for most current software products, the awareness of incorporating security related activities among software developers is being increased rapidly [7]. As a result, there has been an adequate amount of work on integrating security into the software development life-cycle. It is worthwhile to mention that privacy is implicitly integrated in the software development life-cycle, as security considers and adopts privacy requirements during the requirements engineering phase [58]. As a result, the explicit work on integrating privacy into the software development life-cycle is slightly limited.

The Software Assurance Forum for Excellence in Code (SAFECode) published a guide of fundamental best practices for secure software development [96], [97]. The guide was initially released in 2008 covering the areas of: 1) training; 2) requirements; 3) code handling; and 4) documentation [96]. In 2011, the second release of the guide covers the areas of: 1) design; 2) development; and 3) testing [97]. Figure 2.13 summarizes the SAFECode security practices for both releases.

McGraw [75] developed seven software security practices, called Touchpoints to be used during software development life-cycle. The practices help in applying security
### Requirements
- Conducting Risk Assessment
- Identifying Security Requirements
- Defining Security Development Plan
- Prioritizing Security Requirements
- Reviewing Security Requirements

### Technology/Code Integrity & Handling
- Using Current Compiler Toolset
- Using Static Analysis Tools
- Backup and Recovery
- Business Continuity Services
- Physical and Network Security
- Configuration Management Systems

### Design
- Conducting Threat Modeling
- Using Least Privileges
- Implementing Sandboxing

### Programming / Coding
- Minimizing Unsafe Functions Usage
- Validating Input
- Using Canonical Formats
- Avoiding SQL Injection
- Eliminating Weak Cryptography
- Using Robust Integer Operations for Dynamic Memory Allocations *
- Using Logging and Tracing
- Using Anti-Cross Site Scripting (XSS) Libraries
- Analyzing Code via Dynamic Tools
- Reviewing Codes Manually

---

Figure 2.13: SAFECode Security Practices by Simpson [96], [97] (Adapted)
through achieving three main goals: 1) risk assessment (knowing and understanding common risks); 2) security-based design (designing for security); and 3) objective risk analysis and testing. To achieve the three goals, McGraw devised seven practices and integrated them into software development life-cycle. The practices are (as shown in Figure 2.14) : 1) building abuse cases; 2) writing security requirements; 3) conducting internal and external risk analysis; 4) performing risk-based security tests; 5) using static analysis tools to discover common vulnerabilities in coding; 6) performing penetration testing; and 7) monitoring security breaks (attacks).

Microsoft Corporation [58] has developed a seven phases Security Development Life-cycle (Microsoft SDL) based on the traditional software development life-cycle (SDLC). The Microsoft SDL is a collection of 17 security activities applied to software development process in order to get secure software. Each phase has set of practices to be processed and achieved through. Figure 2.15 shows Microsoft SDL along with the practices.
Figure 2.15: The Microsoft SDL [58] (Adapted)
Alkussayer and Allen [7] developed a framework to integrate security patterns and best security practices into the software development life-cycle (SDLC). They collected the best security practices and activities from different sources, such as [96, 75, 58]. Moreover, they collected a set of security patterns from various resources as well. The framework has two component sets, one component set represents a six stage development process of the best security practices, and the other component set represents a four stage utilization process of security patterns. According to the authors, the relationship between the security practices and patterns is bidirectional. Therefore, the integration is applied through interviewing the practices and patterns of each stage. The authors claimed that the integration framework would strongly relate the security patterns with their appropriate security best practices, which leads to clear and simple development for secure software. Also, the integration would help in reducing the ad-hoc security application risks. Figure 2.16 depicts the ISDF framework.

The authors in [7] also claimed that their integration framework would address two concerns when dealing with security patterns. The two concerns are:

- The difficulties of utilizing security patterns.

- The lack of security patterns in some software development life-cycle’s stages.

The framework eliminates the need for utilizing security patterns due to the existence of the practices. It also helps in applying security at the stages that lack of security patterns by using the practices as well.

Despite the continuous research on security, it has not been thoroughly investigated like the other quality attributes. Perhaps this is because security is considered as a qualitative non-functional requirement that is usually considered as an add-on after software product development is completed or at the latest stages. Consequently,
Figure 2.16: The ISDF Framework by Alkussayer and Allen [7]
other quality attributes, such as usability, have become more mature than security [8], especially in terms of measurement.

**Integrating Usability into Software Development Life-Cycle:**

Few research studies propose integrating usability into software engineering development life-cycle [65], perhaps because of the many reasons as Garfinkel listed in [43]. Among the available proposals, some of the proposals come from the HCI field, while other proposals come from the software engineering field. According to Juristo and Ferre [65], neither of the two fields’ proposals lead to a successful integration of usability into software engineering, simply because the HCI proposals are not suitable for software engineering. Additionally, they are not detailed enough to the way that software engineers can understand and make uses of such proposals. On the other hand, the software engineering proposals are not adequately detailed in usability.

Ferre [33] wrote about integrating usability techniques into the software development process. From HCI literature, there are three characteristics of user-centered software development process: 1) active user involvement; 2) clear understanding of user and task requirements; and 3) based on an iterative approach. According to Ferre, the first two characteristics can be met by applying usability activities and techniques, while the third characteristic requires the involvement of the software development life-cycle. To this end, Ferre developed a four-step research process: First, he surveyed the HCI literature to determine the usability activities that are involved in software development process. Three main usability activities have been found to be involved in development process: 1) analysis activities; 2) design activities; and 3) evaluation activities. Second, he surveyed the usability engineering literature to
determine the usability techniques. A number of 82 techniques have been collected, and among those 82 techniques, only 51 usability techniques can be included into software development process as shown above in Figure 2.10. The selection criteria are: 1) alienation to software engineering; 2) general acceptability in the HCI; and 3) cost of application. Third, the author adapted the usability activities to the software engineering life-cycle (SDLC) phases. Three of the SDLC phases are involved: 1) requirements; 2) design; and 3) testing. Fourth, Ferre mapped between the usability activities and the involved phases of the SDLC. Finally, he allocated the usability techniques to the involved SDLC phases activities. Figure 2.17 depicts the above work.

Based on the above research, Ferre has come up with two processes, one for generic development stages and another for increments. The stages in the development process are: 1) elaboration; 2) iterative cycles; and 3) evaluation, as shown in Figure 2.18.

The increments of the incremental software development process are as the following: 1) early analysis; 2) usability specifications; 3) early usability evaluation; 4) regular analysis; 5) integration design; 6) regular usability evaluation; and 7) usability evaluation of installed systems.

As a further work of [33], Ferre et al. [34] proposed a framework for integrating HCI techniques (usability practices) into the software process. They used the same methodology that Ferre used in [33], but in this work they surveyed the HCI literature to determine HCI techniques (95 techniques). They used six criteria to characterize the techniques in order to select the HCI techniques that both improve usability and can be applied into software process as well. The result of their work is a three-views framework: 1) view by HCI techniques; 2) view by development activities; and 3) view
Figure 2.17: Mapping Usability Techniques to Development Activities by Ferre [33](Adapted)
Karat et al. [25] wrote a book chapter on usability in software development lifecycle. In order to apply usability during software development, they attached set of HCI methods to each of the software engineering development phases as follows:

1. Requirements Phase: six HCI methods can be applied to integrate usability during this phase. The activities are:

   • Interviews and surveys of end-user requirements.
   • Focus groups.
   • Field studies of end-user work context.
   • Task analysis.
   • Benchmark studies.
   • Competitive evaluation.

   The following deliverables can be resulted after applying the above activities during the requirements phase:

   • End-user requirements definition.
- User profile definition.
- Core task scenarios.
- Usability objectives specification.

2. Design, Development, and Testing Phases: seven HCI methods can be applied to integrate usability during this phase. The activities are:

- Initial design development.
- Ongoing data from users.
- Prototype development (low, medium, and high fidelity).
- Design walk-through with small groups of users.
- Heuristic evaluations and other types of inspections.
- Usability tests (laboratory or field) with individual users.
- Prototype redesign.

The following deliverables can be resulted after applying the above activities during the three phases (design, development, and testing):

- Validated and usable user interface design.
- Validated user model of the task flow.
- Validated and updated core user task scenarios.
- Validated and updated system requirements document.
- Validated and updated user profile.
- Objective performance data about achievement of usability objectives.
- User interface style guide.
Schneidewind et al. [93] wrote about using personas in user and task analysis during requirements engineering phase. According to the authors, software engineers face three challenges when integrating users’ needs into software projects. The challenges are:

1. Ignoring users’ needs due to lack of clear user description.
2. Lack of mutual understanding of the targeted users by the project team.
3. Users’ tasks and context of use do not match the requirements.

In order to address the above challenges, Schneidewind and his co-workers derived nine hypotheses about the ways that personas could support in requirements engineering activities (elicitation, analysis, specification, and validation). The nine hypotheses are: 1) modeling and prioritizing actors; 2) identifying and illustrating scenarios; 3) prioritizing and illustrating use cases; 4) specifying relationships among actors and use cases; 5) identifying and specifying nonfunctional requirements; 6) illustrating the conceptual model; 7) prioritizing requirements; 8) supporting the requirements specifications; and 9) supporting the approval of the requirements validation. Figure 2.19 maps the activities that personas could support to requirements engineering activities.

Fischer [37] wrote about the international standards-based methods used for integrating usability engineering into software development lifecycle. Fischer stated that despite the still-on-progress work of integrating the two fields, such goal seems to still being challenging. Fischer stated that there are many possible reasons for such challenge. The reasons are:

1. Usability engineering methods and standards have been developed in specific cases, so they become non-transferable to other disciplines.
Figure 2.19: Mapping personas’ supports to requirements activities by Schneidewind et al. [93]
2. No suitable management strategies are currently available to smoothly implement the quality aspects of usability.

According to Fischer [37], the work for integrating usability engineering and software engineering is categorized in four categories: 1) concrete implementation, 2) common specifications, 3) definition of processes and models, and 4) abstract and generic approaches. Among the four categories, Fischer concluded that the category of the definition of processes and models seems promising because it allows for more abstract integration. The first, second, and fourth categories are not recommended by the author because the first and second categories give very specific and detailed integration that might not be transferable to other cases, while the fourth category is too abstract. Figure 2.20 summarizes Fischer’s proposal.
Figure 2.20: Usability Engineering and Software Engineering Integration Proposal by Fischer [37]
2.1.3 Integrating Security and Usability

Software quality attributes in general, and security and usability in particular, are important nonfunctional requirements for most successful computer systems to be achieved [45, 8, 33]. To that goal, many studies been conducted and considerable amount of methods, frameworks, and techniques have been proposed to make such attributes applicable to software development activities, patterns utilization, and evaluation. However, security has not been investigated enough comparing to the other quality attributes [8].

Studies indicate that usability works against security; likewise, security works against usability. This thought has been spread among the majority of usability and security researchers [26, 36]. However, several researchers took the opposite direction, believing that usability and security are not contradictory concepts as explained in the Human-Computer Interaction and Security (HCI-SEC) section.

It is worthwhile to note that, to my knowledge, no work has been conducted that integrates privacy and usability. The possible reasons for the lack of research on such integration have been listed at the beginning of this report (introduction). The next section depicts the intricate nature of the problem of integrating usability and security.

Nature of the Problem of Integrating Security and Usability:

Referring back to the quality attributes section (Section 2.1.1), usability can be summarized in one concept: Ease of Use. In contrast, security can be summarized in one concept: Protection. Therefore, integrating usability and security can then be summarized as integrating ”ease of use” and ”protection.” However, it seems that there is a trade-off between the two concepts. For over 30 years, it has been believed that there
is an inherent conflict between usability and security [26]. The following paragraphs explain the different points of view about the relationship between security and usability.

**Security-Usability Trade-Off:**

Whitten and Tygar [105] stated that usability problems contradict security empowerment efforts. They indicated that the more usable the computer system, the less secure the system is [105].

Adams and Sasse [1] found that in order to have strong password mechanisms, usability must be weakened, as it is difficult to develop strong and usable password generation mechanisms at the same time [1].

Lampson [72] reported that one of the main reasons for security systems' failure in practice is the conceptual semantic problem. Users and administrators do not like security mechanisms in computer systems because they believe that security mechanisms weaken systems' usability. Lampson also reported that software developers look at security mechanisms in computer systems as time consuming efforts that interferes with the functionality of computer systems [72].

Weir et al. [102] conducted an experimental study to compare convenience and usability for three e-banking authentication tokens that each use different authentication methods. Their study showed that there is a trade-off relationship between usability and security from the user's viewpoint. In addition, the study has enhanced the understanding of the trade-off between usability and security [102].

Kainda et al. [67] wrote about balancing usability and security in secure systems. The authors developed a usability-security threat model that identifies usability factors (effectiveness, satisfaction, efficiency, accuracy, memorability, and knowledge/skills) from one side, and security factors (attention, vigilance, conditioning, motivation, social context, memorability, and knowledge/skills) from the other side. Furthermore, the
authors considered two opposite scenarios (usage and threat). The usage scenario analyzed usability through internal and external de-motivators, while the threat scenario was used to analyze security through internal and external motivators. An interesting, but not promising, result reached by the authors is that it is unrealistic to have both usability and security in secure systems; the best that can be achieved is narrowing the usability-security gap [67].

Recently, a rapidly growing group of researchers (HCI-SEC Workgroup) has looked at the issue of integrating usability and security from another angle. Gutmann and Grigg [46] stated that there are several choices to consider when dealing with aligning security and usability [28]. In general, the ways that they look at it is that either through equalizing security and usability, or prioritizing one over the other. The work on both views is examined in the following two sections.

**Equality:**

Equality occurs when usability and security are combined into one concept and considered from the initial moment of developing computer systems.

DeWitt and Kuljis [29] took one HCI-SEC principle, ”the Principle of Least Authority (POLA).” They extended it and proposed a new principle called ”The Principle of Least Authority for Real Internet Security (POLARIS).” They conducted a laboratory experiment based on POLARIS. They concluded that usability and security are better aligned when they go together from the outset of computer systems deployment [29]. Moreover, Balfanz et al. [12] stated that security and usability must be designed simultaneously [29].

**Prioritizing:**

Prioritizing occurs when one of the two terms *usability* or *security* is used to work for the benefit of the other. Obviously, security cannot work for the benefit of usability, because
it is over intricate to generate "Secure Usability Mechanisms." Rather, it is easier to generate "Usable Security Mechanisms." The later approach has been supported by the majority of HCI-SEC researchers.

Flechais et al. [38] wrote a paper an usable security. They used the AEGIS (Appropriate and Effective Guidance for Information Security) to prove that usable security mechanisms can be achieved by considering usability as a security requirement from the beginning of security mechanisms development lifecycle. They concluded that counting usability as a security requirement improved the successfulness of securing computer systems [38].

Saltzer and Schroeder [89] considered usability as one of their eight information protection principles when they added "psychological acceptability" as a security principle, where this simply means "usable security" [26], [89].

Balfanz et al. [12] conducted an experimental study on PKI to find out how usable PKI and other security tools such as SSL and IPSec. They found out that these tools are only applicable when security is considered. However, these are not usable because the usable security was not considered when these tools were built [12].

In conclusion, integrating usability and security is challenging. This fact is stated by most HCI-SEC researchers in previous papers.
2.1.4 Human-Computer Interaction (HCI)

In 1980’s, a group of researchers gathered together and formed a community called Human Computer Interaction (HCI) to share the work of making the interaction between computers and humans applicable, usable, and safe, in order to achieve the goal of such interaction in the best way. Preece et al. [81] quoted the HCI definition as, "human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.” There are three main goals of HCI: 1) usability; 2) safety; and 3) functionality. The HCI group presented approaches and principles that are based on cognitive psychology.

Rogers [87] stated that there are three main approaches for HCI:

- **The Basic Research Approach**: working on main stream information processing.
- **The Cognitive Modeling Approach**: its main focusing on user’s goals achievement.
- **The Diffusion of Popular Concepts Approach**: focusing on users capabilities and limitations.

The two approaches (The Cognitive Modeling and the Diffusion of Popular Concepts) are combined under one approach called ”User-Centered Design Approach” [100, 91]. Figure 2.21 illustrates the concept of using the human-centered approach in the HCI field.

In addition to relying on the user-centered design approach in designing HCI products, the HCI design model has two more promises to rely on. The promises are: integrating knowledge and expertise from the different disciplines of HCI, and making
the design to be highly iterative [81]. HCI specialists follow three processes in order they produce usable computer systems. The processes are [81]:

1. **Understanding** psychological, organizational, social, environmental, and ergonomic factors that play an important role on how effectively people interact with computer technology.

2. **Developing tools** and techniques based on understanding the factors to make sure that computer systems are designed to meet the people’s needs.

3. **Achieving** usability goals of effectiveness, efficiency, and satisfaction; and safety.

Saltzer and Schroeder [89] wrote about information security. They published eight principles for information security, one of which was ”Psychological acceptability.” This principle is known later by the HCI group as ”Usability,” and then by HCI-SEC as ”Usable-Security.” The remaining seven principles are: economy of mechanism, fail-safe default, complete mediation, open design, separation of privilege, least privilege, and least common mechanism [89].

Sasse [91] wrote about the importance of evolving HCI specialists in security mechanisms’ design. She stated that the current available security mechanisms are not
working properly because the users are not interacting with them properly. According to Sasse [91], the cause of the improper interaction with security mechanisms has three reasons:

1. Lack of usability of security mechanisms due to the conflict of interest between the security mechanisms and tasks and work-flow of computer system. This conflict happens because the system’s designers consider security mechanisms during interface building only, but not during the system’s tasks and work-flow determination.

2. Lack of users motivation, as users think that they are not at risk. Moreover, users think they would never be able to stop an attack from happening, even if they follow security rules and policies.

3. Lack of HCI specialists’ involvement in designing security mechanisms.

**Behavior Studies and Evaluation:**

As an abstract view of engineering evaluation, there are four reasons for evaluation [81]: 1) *understanding* the real world; 2) *comparing* designs; 3) *engineering* towards a target; and 4) *checking* conformance to a standard. Working towards each of the four reasons, there are two major types of evaluation to be used: formative and summative. Formative is used during the development processes, while summative is used after product is completely developed [41, 81]. Engineering evaluation is divided into two different classes: analytical and empirical evaluations. Analytical evaluation is used during the early development stages of requirements and early design (before involving users or prototypes). There are many methods for analytical evaluation, such as: heuristic evaluation, cognitive walk-through, usability- expert reviews, or group design review
Empirical evaluation is used during the rest of design and evaluation (testing) phase. There are two main methods for empirical evaluation:

1. **Informal observation**: used during the requirements phase. Moreover, it is used to test prototypes. It is also known as "observing and monitoring usage and interpretive evaluation" [81].

2. **Formal/systematic controlled laboratory studies and comparative surveys and interviews**: This type of empirical evaluation is divided into two methods:

   (a) **Laboratory Studies**: usually used during middle and end of design phase. It is used to test performance, times, accuracy, or errors. Laboratory studies are also known as "experiments and benchmarking" [81].

   (b) **Comparative Surveys and Interviews**: usually used during the evaluation (testing) phase. It is used to test users satisfaction. Surveys and interviews are also known as "collecting users’ opinions" [81].

Moreover, the research studies that apply the analytical evaluation methods; and the informal observation and the comprehensive survey methods of empirical evaluation are usually considered to be qualitative research studies, while the research studies that apply the formal/systematic controlled laboratory methods of empirical evaluation are considered to be quantitative research studies. Qualitative research gives verbal or non-numerical results, while quantitative research gives numerical results through measurable attributes [41]. There are three major strategies or goals of research evidence. These strategies/goals are [41]:

1. Generalization, achieved through surveys.

2. Precision, achieved through controlled laboratory experiments.
3. Realism, achieved through field studies and observation.

**Surveys and Interviews:**

Converse and Presser [23] authored a book on handcrafting the standardized questionnaire that is used in designing surveys and interviews. The authors gave general strategies on how surveys and interviews questionnaires are designed and written. Figure 2.22 illustrates the general strategies.

Some specific empirical findings were discussed in detail along with general lessons. Finally, the two stages (exploration and pretesting) of questionnaire handcrafting were discussed. During the exploration stage, professional experts check the survey questions carefully, and cultural insiders are interviewed properly. Moreover, the authors recommended borrowing questions from others during the exploration stage. The pretesting stage has three major activities:

1. **Pretest Design Strategies:** the way that the pretest is introduced to the respondents and the way that the interviewers act. There are two strategies can be followed:
(a) **Participating**: introducing the pretest to the respondents as "a practice run" and the interviewers ask the respondents to explain their reactions and answers.

(b) **Undeclared Pretest**: introducing the pretest to the respondents as "a real questionnaire" and the interviewer conducts the interview formally.

2. **Purposes of Pretest**: the reason for creating the pretests. There are two goals of performing the pretests:

   (a) **Testing Specific Questions**: through using variation, meaning, task difficulty, and respondent interest and attention.

   (b) **Testing the Whole Questionnaire**: through examining the flow and naturalness of the sections, the order of questions, skip patterns, timing, respondent overall interest and attention, and respondent well-being.

3. **Pretest Phases**: number of pretest steps. According to the authors, the number of the steps performed during a pretest depend on the questionnaire and the number of the participants. However, it is recommended that the number of pretest phases should be at least two along with one evaluation:

   (a) Developmental Pretest I.

   (b) Evaluating Pretest.

   (c) Polishing Pretest II.

**Experiments and Observations:**

Ritter et al. [85] developed a practical guide for the research process of running behavior studies that need human participation. The guide takes the new researchers and research
assistants from the initial stages of human-based research until reporting the results of the research. Figure 2.23 describes an overall road-map for the human participant research studies that Ritter et al. have developed.

Behavior studies can be conducted as observational laboratory experiment using various methods to record the observations on the participants’ behaviors. Some of those methods are [85]:

- **Custom experimental software**: the research team develops the software.
- **Keystroke logger**: for example: Noldus, to record task times, reaction time, and response time.
- **Video recording**: to record multiple details, such as task time, respondent’s attitude.
- **Eye trackers**: for example TobiiX50 [27], to record fixations (pauses) and saccades (movements between two consecutive fixations).
- **Mouse clicks**: to record the clicking location on the interface for accuracy check.

Cutrell and Guan [27] performed a behavior experiment to understand how people search on websites. They used eye-tracking-based measures (pupil dilation fixation information and scan paths) to conduct their experiment. Moreover, a couple of non-gaze-related behavior measures (total time on task and click accuracy) were used as well. Two task types were performed (six tasks for each type as shown in Figure 2.24). The task types are:

1. **Navigational tasks**: finding a particular webpage address by searching through multi-length snippets (short, medium, and long as shown in Figure 2.25) web searches
Figure 2.23: Adapted Human Participants Research Process by Ritter et al. [83, 84, 85]
2. **Informational tasks**: finding particular information by searching through multi-length snippets (short, medium, and long) web searches.

   Each of the 12 tasks of Figure 2.24 was tested on each of the three snippet lengths of Figure 2.24. The total number of tasks per each participant was 36 tasks, where the total number of participants was 22. The experiment gave set of important results as the following:

   - **Time on task**: Overall, navigational search was faster than informational search for all snippets’ lengths. However, navigational search took the longest time on long snippets comparing to searching on short and medium snippets. In contrast, informational search took a shorter time on long snippets in comparison to searching on short and medium snippets.
Click accuracy: Overall, the informational search was more accurate than navigational search for all snippet’s lengths. However, navigational search got the best accuracy on short snippets comparing to searching on medium and long snippets. In contrast, the informational searches received the least accuracy on short snippets comparing to searching on medium and large snippets.

Total result fixated: The navigational search did better than informational search for both short and medium snippets testing, while informational search did better than navigational search in terms of long snippets testing.

Based on the above results, the authors presented the following findings in terms of performance: the best navigational search can be achieved through searching on short
snippets, while the best informational search can be achieved through searching on long snippets.

Bahr and Ford [10] conducted experimental studies to analyze computer non-expert users’ response on pop-ups. The goal of the study is trying to diagnose the reasons for the improper working of pop-ups. The authors conducted two studies to reach their goal: the first investigated tentative expectations regarding eye tracking, while the second study established the reliability of the eye tracking results [10]. The conclusion of the studies stated three simple guidelines are:

1. Be nice (do not force the user to do things).
2. Be polite (do not interrupt the user).
3. Do not be a nuisance (do not repeat, require, or ask for decisions frequently).

2.2 Related Work

2.2.1 Human-Computer Interaction and Security (HCI-SEC)

In 2003, a multi-discipline group of researchers formed a work group called Human-Computer Interaction and Security (HCI-SEC) [5, 38]. This group was formed to bridge the gap between usability and security under the main goal of ”Usable Security” [6]. Figure 2.26 illustrates the area of the usable security. In other words their goal is to come up with usable security mechanisms to secure computer systems. There are three initial main approaches for the HCI-SEC:

1. **First Approach**: building systems that work without considering usability during design, then apply usability by enforcing users to get use the security mechanism.
2. **Second Approach**: building systems with design that encourage users to use the Security Mechanisms inventively and correctly, as usability is applied by encouragement.

3. **Third Approach**: Build systems with learnability concern. Usability is applied by teaching and training users about what they need to know to interact with security mechanisms properly and effectively [26, 36, 43].

Figure 2.26 shows the integration between security and usability that results the interrelated usable security area.

**Usable Security:**

Whitten and Tygar [105] defined *usable security* as ”Security software is usable if the people who are expected to use it:

1. Are reliably made aware of the security tasks they need to perform.
2. Are able to figure out how to successfully perform those tasks.
3. Do not make dangerous errors.
4. Are sufficiently comfortable with the interface to continue using it.” [105]
**Human-Centered Design:**

Among the three initial main approaches, the second ”building systems with design that encourage users to use the security mechanisms inventively and correctly” seems to be the best, as it provides a real usability-security alignment solution [43]. This approach considers both usability and security from the initial stages of software development. The other two approaches are not appropriate solutions, rather, they are like Band-Aids that cover the problem without solving it [43]. Therefore, many proposals and studies have been done to align usability and security based on human-centered design.

**User-Centered Interface Design:**

Whitten and Tygar [105] wrote one of the most famous papers on usability and security. They claimed that the main reason for not having usable security mechanisms is caused by lack of user interface design standards that are appropriate for security mechanisms. As the standards for designing user interface of security mechanisms are the same standards used for other software systems’ user interfaces. Another reason is that no clear and good usability evaluation methods are available. They used PGP 5.0, a famous security mechanism user interface, to prove their claim arguing that such interface is the best candidate. They used two evaluation techniques (cognitive walk-through and laboratory user test) to strength the evaluation results. Based on Whitten and Tygar’s results, the claim is supported; there should be user interface design standards particularly for security mechanisms, rather than the generally available user interface design standards [105].

Sasse et al. [92] wrote about usability problems in password security mechanisms. The core idea of the paper is that users are not the truly weakest link in security mechanisms, rather; lack of usability is the weakest link. They conducted four studies
to determine the causes of password security mechanisms’ failure. They concluded by stating the fact that the reason for those security mechanisms to fail is that they have been designed in the traditional security design approach that ignores the human as a major factor in designing successful security mechanisms. As a result, usability is not considered as a fundamental security requirement.

Braz and Robert [15] wrote a paper about how it is important to consider usability when developing security mechanisms. The authors took the eight golden user interface design principles\(^1\) that were written by Shneiderman [95] in 1998, and tested the traditional password systems against them. The results showed that only two out of the eight principles were applicable. In terms of usability and security, Braz and Robert showed how the remaining six golden rules deal with each of the two terms (usability and security). Moreover, the authors conducted a usability and security comparative analysis on most common authentication methods. The authors concluded that more research is needed to figure out how to incorporate human factors in security mechanisms development process from the beginning [15].

Bahr and Allen [11] examined using polite interaction between humans and security systems. Their idea of polite interaction is based on information processing rather than information ownership. According to the authors, the main reason for security practitioners’ failure in providing security systems that involve end-users occurs because systems are built based on the questionable assumption that security is a primary task and takes precedence on end-users’ priority list. In contrast, Bahr and Allen claim that end-users do not consider security as a primary task, rather, as a secondary task.

\(^{1}\)Ben Shneiderman and Catherine Plaisant wrote the eight golden principles of interface design. The principles are: 1) Strive for consistency; 2) Cater to universal usability; 3) Offer information feedback; 4) Design dialogs to yield closure; 5) Prevent errors; 6) Permit easy reversal of actions; 7) Support internal locus of control; and 8) Reduce short memory load. [95]
Therefore, end-users do not pay attention to security pop-ups. Moreover, end-users always consider security pop-ups frustrating when they interrupt them while performing tasks that they consider as primary. To contribute in providing successful security systems, the authors focus their research on the cognitive state of the user when processing information on computer, while he or she is fully engaged in performing a primary task. To this end, Bahr and Allen presented the following polite interaction guideline [11]:

1. Do not interrupt the users while the user is engaged.
2. If you interrupt the user during a task, it has to be true emergency.
3. Be subtle and allow the user to choose when to engage.

In addition to the above contribution, an initial version of a pay-off utility matrix for polite interaction has been proposed. Figure 2.27 displays the pay-off matrix.

**Security-Usability Involvement from The Initial Development Stages:**
Cranor and Garfinkel [26] wrote a paper about history of usability-security integration problem and the importance of overcoming this problem via designing both usability and security mechanisms together from the beginning of the computer systems development lifecycle. The authors also indicated that the usability-security problem is the main reason for the HCI-SEC working group to first meet in 2003 trying to solve the issue [26]. The HCI-SEC working group proposed a solution of *Usable Security Mechanisms*, and suggested three approaches for applying their solution. The approaches are:

1. Applying usable security by enforcement.
2. Applying usable security by encouragement.
3. Applying usable security by teaching and training.

Balfanz et al. [12] performed an experimental study on PKI usability. They found out that security tools like PKI, SSL and IPSec are just perfect when only security is considered; however, when tested on usability they fail. They listed five important lessons that must be considered in order to have usable security tools [12]. The lessons are:

1. Usable security cannot be added at the end of computer system development.

2. Security tools are just tools, not final solutions.

3. Usability must go together with security mechanisms from the beginning, despite the fact that security mechanisms are built on the lower layers and the usability of those mechanisms are built on the upper layers.

4. Customers must be involved from the beginning.
5. When developing security mechanisms, the thinking and the acting must be mutually local.

Flechais et al. [38] wrote a paper on integrating security and usability into requirement and design phases when building secure software systems. The authors used the Appropriate and Effective Guidance for Information Security (AEGIS) methodology to develop a grid computing system with usability and security concerns. They discovered that AEGIS is useful in designing usable security systems because the usability is considered from the beginning of the development lifecycle. The stockholders were involved from the first step, as the stockholders always look for usability. Figure 2.28 shows the AEGIS.

By involving both nontechnical and technical stockholders from the beginning, the security and usability are both achieved and balanced. The authors’ key point is that usability must be considered as one of security requirements, and the security requirements should be applied on security mechanisms to reveal security problems related to usability. Once such problems are unveiled during development, it is easy to solve them iteratively. As a result, more usable security mechanisms are designed, and secure grid computing systems are achieved [38].

Josang et al. [64] came up with important usability principles to be applied when designing security systems in order to get ”Usable-Security Systems.” They mainly took two principles that Kerckhoffs2 wrote for security, and then they extended and analyzed those two security principles. They generated eight new security usability principles.

---

2Auguste Kerckhoffs wrote six basic principles of a communications security system. The principles are: 1) The system must be practically, if not mathematically, indecipherable; 2) It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience; 3) Its key must be communicable; 4) It must be compatible with the means of communication; 5) It must be portable, and its usage and function must not require the concourse of several people; and 6) Given the circumstances that command its application, the system must be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe [46].
Figure 2.28: The Appropriate and Effective Guidance for Information Security (AEGIS) that Flechais et al. used [25]
(four principles for security action usability, and four principles for security conclusion usability). After generating the eight new principles, they conducted risk assessment on such new principles. The result of the risk assessment was a set of eight vulnerabilities. In addition, they showed two case studies in which the principles are tested to prove the existence of resulted usability vulnerabilities because the principles are not applicable to both cases [64].

Dhamija and Dusseault [30] wrote about seven flaws of Identity Management System (IMS) that are related to usability. The authors claimed that the reason for identity management systems’ failure is using the traditional approach when designing such systems. The approach, however, only focuses on security and complicating it, but does not consider users who use IMS. They concluded by suggesting use of the HCI-SEC approach "user-centered" when designing IMSs, because when the "user-centered design" is used, usability of securing the IMSs (which is usable security) is achieved, too [30].

**Post-Hoc Security Layer:**

Gutmann and Grigg [46] stated that security experts spent the 1990s developing security mechanisms that were not yet needed by computer systems at that time. However, when the need for such security mechanisms became mandatory after the 1990s, users found that security mechanisms built in the 1990s were not useful due to lack of usability. Therefore, security experts had to rethink rebuilding their security mechanisms with consideration to usability.

According to the authors [46], there are several possible choices when usability is considered in building security mechanisms: security and usability should be dealt with equally; security comes first; usability comes first; or security is considered as adds-on
layer and built separately as an independent product. Gutmann and Grigg [46] took the last choice (building separate security mechanisms), called it "Post-Hoc Security," and they recalled Auguste Kerckhoffs’ basic principles of security systems. According to the authors, most of the security mechanisms were built with consideration to the first two principles:

1. Indecipherable, neither practically nor mathematically.

2. Not secret, and cannot be disclosed by enemies.

Instead, security mechanisms should be built with high consideration to the last principle that apply usability (easy to use without mental strain or knowledge).

**Designation Design:**

Yee [107] wrote about the conflict between usability and security and how it affects the security mechanisms effectiveness. Yee indicated that the conflict can be avoided by taking a different approach than the traditional security design approach when designing security mechanisms for computer systems. Yee’s proposed approach is that security mechanisms must be achieved by designation design, not by admonition design. As security by designation does not rely on security decisions made by users, who may not know enough about security [107].

Dewitt and Kuljis [29] conducted an experiment on usability and security to find out whether the two terms are oxymoron in computer systems or not. Their experiment relied on the HCI-SEC principle ”The Principle of Least Authority” (POLA), as they took this principle and extended it to generate the new principle, ”The Principle of Least Authority for Real Internet Security” (POLARIS). The authors conducted a laboratory
Figure 2.29: Sesame by Stoll et al. [99]

The results of their experiment supported the previous HCI-SEC studies that normal users (who do not have enough information about security) are not supposed to make security-related decisions; rather, security-related decisions must be pre-predicated during the design of the security mechanisms through designing usability and security together from the outset of computer systems’ development lifecycle [29].

**User Decision based on Providing Security Information Needed:**

Stoll et al. [99] proposed a ”Usable Security Mechanism” solution based on visualization approach. They created a real time simulation system, called ”Sesame.” Sesame works behind a real system by visualizing the security related interaction when security decisions are needed from the users.
Sesame operates as a knowledge-based security decision through providing visual information about related and contacted processes, networks, and external properties. Also, it gives the needed information to non-security experienced users in order to help them in making right security decisions. The result of the experimental study is that the security decision would be better made when security related processes information is made usable (visualized) [99]. Figure 2.29 shows a snapshot of the Sesame system.

Filtering Users and Data:

Lampson [72] stated that computer security is critical, despite the huge amount of progress is being taken to make computer security perfect enough. According to Lampson, security is not about perfection; rather, security is about policy and isolation. Policy is a part of security that users and administrators can understand and share setting it up. Policy can be modeled so users’ and administrators’ view of security become clear. Isolation is a part of security that developers and security experts can apply a security policy, and control the isolation via the policy. Isolation can be achieved via two steps: 1) filtering input data (safe and dangerous); and 2) filtering users (accountable and unaccountable). When these two parts of security (policy and isolation) are properly incorporated, the goal of Usable Security can easily be achieved [72].

Using Biometrics:

Riley et al. [82] did a cross-cultural survey about acceptance of using biometrics authentication technologies in three culturally different countries (the U.K., India, and South Africa). They found that culture has a strong impact on users’ level of acceptance of using the above technologies. The result showed that Indians have the highest level of acceptance, the British have the lowest level of acceptance, whereas South
Africans are in the middle. According to the authors, the reason for that is differences in the degree of concern about privacy and the degree of trust of using technology. Finally, they compared their results with Hofstede’s cultural dimensions and theories of technology acceptance. They claimed that Hofstede’s model does not provide as sufficient explanation as studying contextual issues of each country. This study has brought the role of cultures in usability when using biometrics as security authentication technologies [82].

Fernandez-Saavedra et al. [32] have developed an evaluation methodology to analyze usability factors of biometrics when used in security applications. The methodology focused on analyzing usability factors that affect biometric performance. According to the authors [32], the usability factors that most affect biometrics’ performance are:

- Ease of presentation of biometric characteristics.
- Ease of user-system interaction.
- Ease of learning process.

The methodology examined fingerprints, and proved that it is a useful evaluation methodology for biometrics performance usability factors [32].

Kukula et al. [70] provided an evaluation method for biometric performance usability measurements effects as shown in Figure 2.30. They used the evaluation methodology on a framework (Figure 2.31) that they developed through focusing on generating additional focused measures from traditional system-level evaluation metrics: the failure-to-acquire (FTA); the failure-to-enroll (FTA); the false-accept (FAR); and the false-reject (FAT). The authors claimed that using the above metrics for
evaluation is insufficient to evaluate the usability of biometric performance. To address this issue, they generated the following measures:

- False-interaction (FI).

- Failure-to-detect (FTD).

- Concealed-interaction (CI).

The authors of [70] claimed that the newly generated metrics improved the biometric performance evaluation because the new metrics analyze the interaction between humans and biometric sensors in a more accurate way [70].

Alharby et al. [2] wrote a paper on users’ acceptance of secure biometrics authentication. The authors based their study on one biometric trait, the fingerprint, to determine the factors that affect users’ acceptance to such technology in Saudi Arabia. The findings of their experiment were measured based on the Unified Theory of Acceptance and Use of Technology (UTAUT). The authors added two new factors to the UTAUT (self-efficiency and biometric system characteristics). Figure 2.32 displays the adapted UTAUT model that Alharby et al. developed.
The analysis of their data considered three moderating variables: age, gender, and education level. The results of the experiment finding analysis showed that the two added factors, self-efficiency and biometric system characteristics, to the UTAUT can raise users’ acceptance of secure biometrics authentication as depicted in Figure 2.33.

Kumar [71] conducted a survey study to evaluate usability of alphanumeric password based security systems. The author concluded that alphanumeric passwords cannot be both usable and secure at the same time because the HCI studies had proven that the human brain memory, by its nature, is limited to deal with alphanumeric passwords. Therefore, the author suggested a couple of alternatives, that are:

1. Graphical passwords.

2. Biometrics (fingerprint).
Figure 2.32: The Adapted UTAUT Model by Alharby et al. [2]

<table>
<thead>
<tr>
<th>Variable constructs</th>
<th>Composite reliability</th>
<th>Cronbach's alpha(s)</th>
<th>Average variance extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance expectancy (PE)</td>
<td>0.954</td>
<td>0.887</td>
<td>0.512</td>
</tr>
<tr>
<td>Effort expectancy (EE)</td>
<td>0.937</td>
<td>0.758</td>
<td>0.543</td>
</tr>
<tr>
<td>Social influence (SI)</td>
<td>0.892</td>
<td>0.760</td>
<td>0.689</td>
</tr>
<tr>
<td>Behavioural intention to use (BI)</td>
<td>0.921</td>
<td>0.885</td>
<td>0.596</td>
</tr>
<tr>
<td>Self-efficacy (SE)</td>
<td>0.887</td>
<td>0.844</td>
<td>0.614</td>
</tr>
<tr>
<td>Biometric system characteristics (BSC)</td>
<td>0.863</td>
<td>0.741</td>
<td>0.633</td>
</tr>
</tbody>
</table>

Figure 2.33: Experimental Results by Alharby et al. [2]
Toledano et al. [100] conducted a user-centered usability evaluation study of biometric systems. Their study focused on the three standard goals of usability (effectiveness, efficiency, and satisfaction). Three biometric traits have been chosen for the evaluation study: signature, fingerprint and voice. Figure 2.34 shows how Toledano et al. related the objective and subjective measures.

Using both objective measures for effectiveness and efficiency, and subjective measures for satisfaction as shown on Figure 2.34, the result of their experiment was the following:

- When the three traits were tested and analyzed for effectiveness, the fingerprint was determined to be the best; voice was the second; and finally signature was the worst.

- When the three traits were tested and analyzed for efficiency, the fingerprint was the best in reliability; signature was the best in speed; however, the voice did not provide any clear indication because it was in the middle.
• When the three traits were tested and analyzed for satisfaction, fingerprint was the best, voice was the second, and signature was the worst.

Overall, fingerprint was shown to be the best in terms of usability [100].

Using Principles and Patterns:

Garfinkel [43] provided a collection of principles and invented patterns as a way to integrate the security-usability aliment into systems development. According to Garfinkel, "usability and security can be made synergistic by redesigning systems with specific principles and through the adoption of well-defined patterns" [43]. To this goal, he collected a set of six principles and used them as foundations to came up with 21 patterns. The principles and the patterns can be used to achieve the usable security for computer systems. For documenting the principles, Garfinkel used a template consisting of four characteristics:

1. **Principle**: the name of the principle.

2. **Intent**: the goal to be accomplished from the principle.

3. **Motivation**: the reason for the principle to be important.

4. **References**: further information about the principle.

The six principles that Garfinkel collected are listed on Figure 2.35.

Figure 2.36 shows one example of the principles collected by Garfinkel.

For documenting the patterns, Garfinkel used a template consisting of the following 10 characteristics:

1. **Pattern name**: the name of the pattern.
Figure 2.35: The Six Usable Security Principles by Garfinkel [43] (Adapted)

<table>
<thead>
<tr>
<th>Principle</th>
<th>Consistent Meaningful Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intent</strong></td>
<td>Prevent confusion by using words consistently to convey the same idea or concept in different programs and contexts. Likewise, prevent confusion by assigning consistent meanings to the same word in different applications or contexts.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Technologists in general and computer security practitioners in particular are generally loose with the words used to represent terms and ideas. Different words used for the same idea confuse users, who look for meaning in the differences and frequently create incorrect explanations for the sloppiness. The sloppiness can negatively affect implementations when programmers become confused.</td>
</tr>
</tbody>
</table>

Figure 2.36: A Sample Principle from Garfinkel’s Collected Principles [43]
2. **Intent:** the goal to be accomplished from the pattern.

3. **Motivation:** the reason for the pattern to be important.

4. **Image:** an image to depict the pattern usage.

5. **Applicability:** relevant situations where the pattern can be used.

6. **Participants:** a list of other patterns that the pattern depends on (if presented in *italic*), or individuals who are responsible for applying the pattern.

7. **Implementation:** the way(s) that the patterns can be implemented.

8. **Results:** the expected results of applying the pattern.

9. **Known Uses:** examples of real-life applications or systems that use the pattern successfully.

10. **References:** further information about the pattern.

The 21 patterns that Garfinkel generated are divided into three main groups based on the intentional target of using them. The three groups are:

1. **User Visibility and Sanitization:** ensures elimination of the hidden information that remains on hard drives after a normal delete operation is committed.

2. **Identification and Key Management:** provides solution to secure messaging problem.

3. **Promoting Overall Secure Operation:** techniques and practices to further promote overall secure operations without negative effects on usability.

Figure 2.37 presents each group along with its patterns. Figure 2.38 shows one example of the patterns generated by Garfinkel.
Figure 2.37: The 21 Usable Security Patterns by Garfinkel [43](Adapted)
### Complete Delete

<table>
<thead>
<tr>
<th>Intent</th>
</tr>
</thead>
</table>
| Ensure that when the user deletes the visible representation of something, the hidden representations are deleted as well.  

<table>
<thead>
<tr>
<th>Motivation</th>
</tr>
</thead>
</table>
| Frequently deleting information does not erase all of the copies in the computer; hidden data remains from which the user’s desire to erase information can be subverted. Complete Delete ensures that information that is deleted cannot be recovered.  

<table>
<thead>
<tr>
<th>Applicability</th>
</tr>
</thead>
</table>
| - Removal of text from within documents  
| - Removal of records from databases  
| - File deletion  
| - Erasure of passwords and cryptographic keys in memory  

<table>
<thead>
<tr>
<th>Participants</th>
</tr>
</thead>
</table>
| Delayed Unrecoverable Action  

<table>
<thead>
<tr>
<th>Implementation</th>
</tr>
</thead>
</table>
| Complete Delete is implemented by determining what information stored in the computer system corresponds to the user’s notion of the object being deleted, then overwriting the storage media that holds that information so that the data cannot be recovered. While Complete Delete cannot be implemented for information that is stored offline, the results of Complete Delete can be achieved by encrypting offline information and then using Complete Delete to erase the encryption key.  

<table>
<thead>
<tr>
<th>Results</th>
</tr>
</thead>
</table>
| Prevents forensic analysis from being able to recover information that has been intentionally deleted. Forces designers and organizations to clearly articulate their strategy for maintaining backups and who has access to that information.  

<table>
<thead>
<tr>
<th>Inferred Uses</th>
</tr>
</thead>
</table>
| Apple implements Complete Delete, albeit poorly in the MacOS 10.3 “Secure Empty Trash” command. Microsoft’s Cipher.exe command can be used to overwrite slack space. Both of these implementations have profound implementation flaws and usability problems (see Chapter 3).  

**References**: Chapter 3 discusses how the failure of Complete Delete at the file level has frequently exposed confidential information; Section 4.2 on page 155 shows how problems in Microsoft Word and Adobe Acrobat have resulted in similar disclosures.

---

Figure 2.38: A Sample Pattern from Garfinkel’s Generated Patterns [43]
2.2.2 Security Assessment

OWASP Risk Rating Methodology:

The Open Web-Application Security Project (OWASP, 2013) [78] developed a subjective security Risk Rating Methodology (RRM) to estimate the security related risks (see Figure 2.39). The approach has a standard risk model to be used at the end of sequence steps. The standard model is based on calculating factors for the likelihood and the impact in order to estimate how severe the risk is as follows:

\[ Risk = Likelihood \times Impact \]  

There are four primary steps used to estimate the security risks for an application, and there are two additional secondary steps used for decision making and customizing the methodology factors. The steps are:

1. Identifying a Risk
2. Factors for Estimating Likelihood
3. Factors for Estimating Impact
4. Determining Severity of the Risk
5. Deciding What to Fix

6. Customizing Your Risk Rating Model

The OWASP RRM is related to our project because it assesses security risks. It is customizable, comprehensive, and easy to apply. However, the assessment methodology is subjective and only provides five levels of severity (note, low, medium, high, and critical), where note level means that there is no severity and critical means that it is a severity level that needs an immediate solution. Moreover, it only estimates one quality attribute without considering the impact of interrelating security with the other quality attributes such as usability.

2.2.3 Usability Assessment

SALUTA Attribute Preference Table:

Folmer et al. (2003) [40] have developed an evaluation methodology called "Scenario-based Architecture Level Usability Assessment (SALUTA).” The evaluation methodology measures the usability application on software architecture during the design phase in order to investigate whether the architecture meets the usability requirements that are previously written during the requirements phase of the SDLC. The method has five main steps:

1. Determining the goal of the assessment (goal selection). This step has three subsequent steps to perform:

   (a) Predicting the level of usability on the software architecture.

   (b) Conducting usability risk assessment (analysis) on the architecture in order to detect usability problems that need to be addressed.
(c) Comparing two software architectures to select the most appropriate one that perfectly goes with usability.

2. Describing the required usability through creating a usage profile suitable for architectural assessment. This step has a set of subsequent steps:

(a) Defining scenario for usability consisting of definitions of the users, the tasks, and the context of operation.

(b) Creation of Attribute Preference Table (APT) to relate scenarios and usability by providing rating value (between 1 and 4) for each adopted usability property (satisfaction, learnability, efficiency, and reliability) on each scenario. The authors stated that such rating technique can also be used as a usability requirements assessment methodology.

(c) Selecting and prioritizing the scenarios based on the APT in order to assess them.

3. Describing the usability related software architectures by providing diagrams, design decision documents, or use case maps (UCM).

4. Evaluating and prioritizing the scenarios based on three criteria:

(a) Number of patterns supporting usability.

(b) Number of design decisions supporting usability.

(c) Amount of behavioral and structural aspects supporting usability from the UCM.
5. Interpreting the selected (optimal) scenario ends the SALUTA process. This step takes the selected scenario and applies it to the software architecture to see if the scenario is able to integrate with the other software quality attributes.

The above research directly feeds into and interrelates with our research work towards usable-security in two points. The first point is related to the first step of the SALUTA, as the first task is supposed to predict the level of usability on the software architecture. However, predicting the level of usability should be based on the usability requirements of the requirements engineering phase. The second point is that the attribute preference table (APT) can be used to assess the usability requirements in order to provide predictive assessment that can be used as a baseline for the architecture evaluation.

2.2.4 Usable-Security Assessment

Design Problems:

Braz et al. (2007) [16] proposed a usability and security design model that relates usability and security problems. The model has four main steps as follows:

1. Creating Scenarios:

   (a) Task Scenarios: conducting the Context of Use analysis to determine the characteristics of the user, the tasks, and the contexts of use (environments).

   (b) Usability Scenarios: finding usability related problems that users can cause or face when performing a task from the task scenarios.
(c) Security Scenarios: finding security related problems (tangible and intangible) that users can cause or interact with when performing a task from the task scenarios.

2. The security problems are related to seven measurable criteria (minimal action, minimal memory load, operability, privacy, security, load time, and resource safety).

3. Then the security problems are related to the nine usability factors (efficiency, satisfaction, productivity, learnability, safety, trustfulness, accessibility, universality, and usefulness).

4. The task scenarios are measured based on the usability criteria that is related to them. The measuring metrics used are: percentage of task completion, ratio of task successes to failure, frequency of program help usage, time spent dealing with program errors, and the number of on-screen UI elements.

In the same paper, Braz et al. [16] proposed a novel metrics-based usability inspection design to design, inspect, and evaluate the usability of security systems (usable-security). The model is called Security Usability Symmetry (SUS). The SUS has a set of steps to be followed that at the end identifies problems that are related to both usability and security together during the design phase. The steps are as follows:

1. Creating Scenarios

   (a) Task Scenarios: conducting the Context of Use analysis to determine the characteristics of the user, the tasks, and the contexts of use (environments).

   (b) Usability Scenarios: finding usability related problems that users can cause or face when performing a task from the task scenarios.
(c) Security Scenarios: finding security related problems (tangible and intangible) that users can cause or interact with when performing a task from the task scenarios.

2. The usability problems are rated as (major, intermediate, or minor) severity levels based on three aspects: frequency, consequences, and persistence of the problem.

3. The security problems are rated as (high, medium, or low) severity levels based on six aspects: authentication, confidentiality, integrity, non-repudiation, access control, and availability.

The above research is directly related to our research in two points. First, the research provides a subjective assessment for both security and usability separately, which does not objectively consider the integration between the two attributes. Moreover, it provides neither subjective nor objective assessment of the integration between security and usability. Second, the model discovers the security-usability problems during the design phase, where the discovered problems were created during the requirements engineering phase. By using this model, problems can be avoided once the integration and the assessment is done during the requirements phase.

Challenge Questions:

Just and Aspinall (2009) [66] proposed a choice-based security and usability assessment method for challenge questions. The study assessed generating, choosing, and answering challenge questions that usually appear when passwords are forgotten by end-users. The assessment method consists of two models to be applied and then analyzed. The models are: the security model to evaluate the security levels for the challenge questions from the attackers’ viewpoints, and the usability model to evaluate usability
for the challenge questions from the users’ viewpoints. The following describes and analyzes both of the models.

The security model uses three attacking methods to evaluate the challenge questions: blind guess, focused guess and observation. The blind guess adopts attacking techniques such as brute force, and dictionary without considering any other factors. The focused guess analyzes the challenge question and then guesses the most likely answers without considering any relation between the question and the user who answered it. For example, when asking for a user’s last name, the focused guess considers all the available last names. The observation considers the publicly available users’ information through users’ accounts, social networks, and physical networks (friends and family members).

The blind and focused guess attacks incorporate the calculated entropy of the possible answers, and provide a security level as follows:

**Low:** The possible answers are less than \(2^{34}\).

**Medium:** The possible answers are between \(2^{34}\) and \(2^{48}\).

**High:** The possible answers are greater than \(2^{48}\).

The observation attack relies on the availability of the user information; therefore, the authors offered other measures to provide a security level, as follows:

**Low:** The possible answers are publicly available to everybody.

**Medium:** The possible answers are known by family members and friends.

**High:** The possible answers are only known by the user.
Table 2.1: The Results of Assessing the 180 Questions on the Three Attacks

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blind Guess</td>
<td>174</td>
<td>4</td>
<td>2</td>
<td>180</td>
</tr>
<tr>
<td>Focused Guess</td>
<td>167</td>
<td>0</td>
<td>13</td>
<td>180</td>
</tr>
<tr>
<td>Observation</td>
<td>124</td>
<td>54</td>
<td>2</td>
<td>180</td>
</tr>
</tbody>
</table>

Each user is asked to provide three challenge questions and write answers to these questions. The security analysis has been done in two steps according to the above levels as follows:

1. The first step is the question assessment: Each collected question is assessed via each of the three attacking methods. The results of the first step are shown in Table 2.1.

2. The second step is the user assessment: A gradual analysis is conducted on each user’s questions through assessing the security level on the blind and focused attacking methods accumulatively (the first question only, the first two questions together, all of the three questions), while the observation attacking method followed a different assessment strategy:

   **Low:** If each of the three questions was evaluated as low.

   **Medium:** If at least one of the questions was evaluated as medium.

   **High:** If at least one of the three questions was evaluated as high.

The authors concluded that having more than one challenge question together (i.e. three challenge questions) improved the overall security level against all of the three attacking methods with respect to the change in the measuring criteria for the observation.
The usability model uses two assessment types: self-assessment by the users themselves and analytical assessment by the researchers. Four usability criteria were used on both of the assessments to evaluate usability. The authors concluded that the challenge questions have usability problems related to memorability, honesty, repeatability, and incidents change.

This work has a relation to our research at three points. First, it assesses usability and security on choice-based challenge questions for authentication. Second, despite assessing the two quality attributes, the work does not assess the relationship between security and usability. Third, security has been assessed subjectively through using the three severity levels.

2.2.5 Usable-Security Authentication Design

Design Guidelines:

Payne et al. (2008) [79] wrote about the history of usable-security. The authors stated that the two most important concerns the usable-security group try to address are authentication and email encryption. They reviewed the literature of both fields to summarize the history of each. Authentication is mostly concerned with passwords. Throughout the authentication history, there have been six types of passwords: passphrases; pass-algorithms; user-friendly passwords; cognitive passwords; graphical passwords and pass-faces; and finally pass-points. The authors recalled Yee’s usable-security design guidelines [106]:

1. Path of least resistance.

2. Active authentication.
98

3. Revocability.

4. Visibility.

5. Self-awareness.

6. Trusted-path.

7. Expressiveness.

8. Relevant boundaries.

9. Identifiably.

10. Foresight.

The authors then showed two successful and two failure design examples where Yee’s guidelines were used as evaluation criteria (see Figure 2.40).
Object-Based Authentication Password:

Biddle et al. (2011) [14] proposed a password generating scheme called Object-Based Password (ObPwd). The scheme allows the users to select an object (e.g., a pdf file) from user’s personal contents and hashes that object using the SHA-1 hashing technique. The result of the hashing is converted into an alphanumeric password that does not have to be memorized via a function called Hash2Text. The authors claimed that the ObPwd provides both security and usability simultaneously (usable-security). According to the authors, the scheme provides usability because the user does not have to memorize the alphanumeric password; rather, the user has to memorize the selected object in order for the ObPwd to generate that password.

This work is related to our research since the scheme is developed as a usable-security design solution. However, the scheme has three negative points as follows:

1. The scheme itself does not increase the security level because the selected object can be corrupted or deleted.

2. The security can be broken once the selected object is known by illegitimate parties.

3. The ObPwd is necessary whenever the password is needed.

2.2.6 Hybrid Approaches

Usability and Security Practices and Activities

Hausawi and Mayron [54] contributed to integrating security and usability into the Software Development Life-Cycle (SDLC). They adapted the Microsoft Guidance of Patterns and Practices for Security Engineering (Microsoft GPPSE) and Microsoft
Security Development Life-cycle (Microsoft SDL) [58]. First, they merged the guidance and the SDL, and then extended the work by including Usability Engineering practices and techniques collected from [24, 33, 25, 95, 81].

As a result, they provided a unified security and usability guidance based on the SDLC. The guidance can be used by software engineers to apply both security and usability during the software development process. Furthermore, such unified guidance is a starting point for further research in aligning security and usability in order to develop a usable-security guidance. Table 2.2 displays the unified security and usability guidance for the software development process.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Security Practice</th>
<th>Usability Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Security training</td>
<td>Human-centered</td>
</tr>
<tr>
<td>Requirements</td>
<td>Security objectives</td>
<td>Context specification</td>
</tr>
<tr>
<td></td>
<td>Security requirements</td>
<td>Usage scenarios</td>
</tr>
<tr>
<td></td>
<td>Quality gates</td>
<td>User analysis</td>
</tr>
<tr>
<td></td>
<td>Risk assessment</td>
<td>Task analysis</td>
</tr>
<tr>
<td></td>
<td>Specifications</td>
<td>Usability specifications</td>
</tr>
<tr>
<td>Design</td>
<td>Attack surface analysis</td>
<td>Concept development</td>
</tr>
<tr>
<td></td>
<td>Threat modeling</td>
<td>Prototypes</td>
</tr>
<tr>
<td></td>
<td>Design review</td>
<td>Interaction design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Design review</td>
</tr>
<tr>
<td>Implementation</td>
<td>Approved tools</td>
<td>Approved tools</td>
</tr>
<tr>
<td></td>
<td>Security patterns</td>
<td>User interface patterns</td>
</tr>
<tr>
<td></td>
<td>Static analysis</td>
<td>Interface development</td>
</tr>
<tr>
<td></td>
<td>Code review</td>
<td>User interface review</td>
</tr>
<tr>
<td>Testing</td>
<td>Dynamic analysis</td>
<td>Expert review</td>
</tr>
<tr>
<td></td>
<td>Attack surface review</td>
<td>Usability evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acceptance testing</td>
</tr>
<tr>
<td>Deployment</td>
<td>Incident response plan</td>
<td>Surveys and interviews</td>
</tr>
<tr>
<td></td>
<td>Security review</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Release Archive</td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td>Incident plan</td>
<td>Usability review</td>
</tr>
</tbody>
</table>
Usable-Security Principles and Patterns

Mayron et al. [74] contributed to integrate security and usability based on the user-centered design approach through applying usable-security when using biometrics authentication systems for Universal Access (UA) purposes. Biometrics are increasingly used as proper solutions to achieve both security and usability for UA. However, biometrics are vulnerable to poor usability despite their strength in achieving security, so Mayron et al. recommend using the principles that Garfinkel [43] collected to apply usable-security for biometrics-based UA through enhancing biometrics usability and leverage security at the same time. Figure 2.41 shows that usable-security lies at the intersection of security and usability and works toward achieving their goals.
Chapter 3

Usable-Security Assessment during Requirements Engineering

Preface

The balance between security and usability must be addressed as early as possible in the Software Development Life Cycle (SDLC) to ensure the inclusion of usable-security in software products. Unfortunately, there has been little research on assessing and integrating security, usability, and usable-security during the requirements engineering phase of the SDLC. To address that deficiency, this chapter proposes an Assessment Framework for Usable-Security (AFUS) based on two well-known techniques from the decision science field.
3.1 Introduction

Security and usability are two important software quality attributes that should be incorporated into software projects during the requirements phase [33, 45]. However, implementing both in a particular product is problematic because the goals of security and usability are often in conflict [1, 29, 102, 104]. Much research has been done by HCI and security specialists to bring security and usability into a synergetic integration [26, 72] and a more recent approach to resolving these potential conflicts is to employ a hybrid attribute, namely usable-security [43, 54, 74]. However, most of the research on usable-security has focused on the design phase of the SDLC, resulting in a usable-security assessment gap in the requirements phase [21]. A recent literature survey found no current usable-security assessment methodology that addresses the requirements phase.

The field of Decision Science provides tools and techniques for resolving conflicts between differing objectives [22]. In this chapter, we propose an Assessment Framework for Usable-Security (AFUS) that explores the benefits of using two well-known techniques from Decision Science, namely Utility Functions and Decision Trees, for assessing the balance between security, usability and usable-security represented in the set of requirements for a particular software product.

The goal of this work is not to produce an objective measure for comparing two products, but rather to generate a metric that developers can use to gauge the balance between the attributes. We assume that the developers of a product are aware of the balance between security and usability that is appropriate for their product, thus the proposed technique is intended to assist in reaching that desired balance. As changes to the requirements are made, reassessment using AFUS can indicate if the product
has shifted to a greater emphasis on one attribute at the expense of the others, or if all attributes have moved towards the developer’s preferred equilibrium. The work of this chapter can either be considered as a component of the Usable-Security Framework, or can be used as a stand alone assessment and balancing mechanism in order to gauge the balance between the attributes. In the case where this chapter is used as a component of the framework, it is related to Chapter 4, as the results of assessing the attributes’ requirements from this chapter can be used to identify the targeted priority sensitivity levels in Chapter 4.

Section 3.2 introduces our usable-security assessment framework (AFUS). Section 3.3 discusses the results, and Section 3.4 concludes this chapter.

### 3.2 Assessment Framework for Usable-Security (AFUS)

The proposed framework (see Figure 3.1) has three main components: 1) requirements filtering and merging using the OWASP Risk Rating Methodology [78] for security and the SALUTA Attribute Preference Table [40] for usability as guides; 2) utility functions; and 3) a decision tree.

The framework was evaluated by using case studies based on three real world scenarios. Scenarios are often used in software engineering to gather and validate non-functional requirements [21] and in HCI to improve communication between stakeholders and developers [9]. The first scenario was based directly on the non-functional usability, humanity, and security requirements from the Volere Requirements Specification Template [86]. Two additional scenarios were produced with different specifications for their non-functional requirements. The usable-security requirements
Figure 3.1: The Assessment Framework for Usable-Security (AFUS)

were derived from the other non-functional requirements, as the Volere template does not have a usable-security section.

### 3.2.1 Requirements Filtering and Merging

Requirements engineering is the first phase of the SDLC [98]. In this phase, the stakeholders meet to establish the project requirements and analyze the generated requirements [98]. Among those requirements are security and usability requirements. The scope of the AFUS is limited to the requirements that are related to security, usability, and usable-security. Therefore, the intended system requirements are filtered to select and gather the requirements that are within the scope of the AFUS, and then they are grouped into three main groups, security requirements, usability requirements, and usable-security requirements. After making the three groups, their requirements
are rated based on a unified five-value rating guidance: [1 - Not Important], [3 - Some Important], [6 - Important], [7 - Very Important], and [9 - Critical]. It is important to state that the rating guidance provides clear differentiation (border-line) between some important and truly important requirements through abandoning two values from the rating scale in order to make the differentiation distinguishable enough. The requirements of each of the three groups are rated according to their importance as the following processes:

**Security Requirements Group (SR).**

After adapting OWASP Risk Rating Methodology (OWASP RRM) [78], the adapted methodology is used as the foundation to assess security requirements. The importance of each security requirement is rated on each security property [80]: confidentiality (CI), integrity (NI), and availability (AI), based on Table 3.1. We observed that OWASP does not use standardized rating values, as each security property has its own rating values. Also, there are some gaps in rating some properties. For example, the OWASP uses the values (1, 5, 7, and 9) to estimate the impact of availability loss on the system if vulnerability is exploited. Such estimation values may experience lack of accuracy because the estimated impact between 1 and 5, let us say 3, cannot be accurately given. This may force the estimator to estimate as either 1 or 5. Therefore, we adapted the OWASP rating methodology to fill in the rating gaps and make it standardized and unified in order to utilize it for the usability requirements with respect to security requirements, as will be explained later in Section 3.2.1. The security rate of each requirement ($SecR$) is calculated through averaging the rates of the three security
properties. The calculation equation is shown as follows:

$$SecR_i = \frac{CI_i + NI_i + AI_i}{3}$$ (3.1)

The overall security rate ($SEC$) for the requirements set is derived by:

$$SEC = \sum_{i}^{SR} SecR_i$$ (3.2)

Moreover, to assess the shares ($SH$) of each security property, confidentiality ($SH_c$), integrity ($SH_n$), and availability ($SH_a$), summing of each property importance is calculated from all of the security requirements and divided by overall security rate ($SEC$) as depicted on the following three equations respectively:

$$SH_c = \frac{\sum_{i}^{SR} CI_i}{SEC}$$ (3.3)

$$SH_n = \frac{\sum_{i}^{SR} NI_i}{SEC}$$ (3.4)

$$SH_a = \frac{\sum_{i}^{SR} AI_i}{SEC}$$ (3.5)

**Usability Requirements Group (UR).**

SALUTA [40] is a usability assessment technique used to rate usability based on assigning quantitative values for usability preferences. An adapted rating methodology based on SALUTA Attribute Preference Table [40] is used to rate usability requirements,
as the same rating values (between 1 and 9) of the adapted methodology used to rate security requirements group is used to rate the importance of each usability property [59], effectiveness (EI), efficiency (FI), and satisfaction (SI), in order to rate each usability requirement. It is worthwhile to note that having unified rating values for both security and usability requirements provides a consistent qualification strategy used to measure the requirements. Table 3.1 is used as a guidance for the rating process. A five-value rating is used (1, 3, 6, 7, and 9). This rating can be justified as the most appropriate for the usability requirements rating process because it efficiently helps with rating any usability requirement on the three usability properties and the rater would not be forced to give an inappropriate rate. SALUTA uses a four-value rating (1, 2, 3 and 4), as it ranks scenarios based on four usability properties (each property gets one ranking value). Our framework rates requirements, but does not rank them, and the rating guidance should work perfectly with all of the requirements. The usability rate of each requirement ($UsaR$) is calculated through averaging the rates of the three usability properties. The calculation equation is as follows:

$$UsaR_j = \frac{EI_j + FI_j + SI_j}{3}$$

(3.6)
The overall usability rate \( (USA) \) for the requirements set is derived by:

\[
USA = \sum_{j}^{UR} UsaR_j
\]  

(3.7)

To assess the shares \( (SH) \) of each usability property, effectiveness \( (SH_e) \), efficiency \( (SH_f) \), and satisfaction \( (SH_s) \), summing of each property’s importance is calculated from all of the usability requirements and divided by overall usability rate, \( USA \), as depicted in the following three equations respectively:

\[
SH_e = \frac{\sum_{j}^{UR} EI_j}{USA}
\]  

(3.8)

\[
SH_f = \frac{\sum_{j}^{UR} FI_j}{USA}
\]  

(3.9)

\[
SH_s = \frac{\sum_{j}^{UR} SI_j}{USA}
\]  

(3.10)

Hence, after rating the requirements of both security and usability based on the OWASP RRM [78] and the SALUTA APT [40] respectively, it resulted in overall static ratings for both attributes \( (SEC \) and \( USA \). A static assessment is calculated for the two attributes \( (SEC_{static} \) and \( USA_{static} \) by applying the following equations:

\[
SEC_{static} = \frac{SEC}{SEC + USA}
\]  

(3.11)

\[
USA_{static} = \frac{USA}{SEC + USA}
\]  

(3.12)
### Table 3.2: Initial Usable-Security Importance Rating Guidance

|----------------|--------------|--------------------|---------------|-------------------|-------------------|

**Usable-Security Requirements Group (USR).**

This requirements group has two sub-groups: 1) initial usable-security requirements sub-group \((IUSR)\); and 2) merged usable-security requirements sub-group \((MUSR)\).

The overall usable-security rate \((USA_{SEC})\) is calculated by summing the two sub-groups. The next sections describe each sub-group.

**Initial Usable-Security Requirements Sub-Group (IUSR).** The requirements of this sub-group are rated in different rating methodology, as usable-security does not have standard properties like security and usability do. In addition, one usable-security requirement may mix security and usability properties. Moreover, the requirements that are based on the Human-Computer Interaction and Security (HCI-SEC) are considered as IUSR \([30, 36]\). Therefore, each initial usable-security requirement is rated based on Table 3.2, then multiplied by 2, and then divided by 3 as illustrated in this equation:

\[
IUsa_{Sec}R_k = \frac{I_k \times 2}{3} \tag{3.13}
\]

The overall initial usable-security rate \((IUSA_{SEC})\) is calculated from the following equation:

\[
IUSA_{SEC} = \sum_{k}^{IUSR} IUsa_{Sec}R_k \tag{3.14}
\]
Merged Usable-Security Requirements Sub-Group (MUSR). In order to assess usable-security in the best appropriate manner, both security and usability requirements must be analyzed properly and carefully with merging and alignment intention in mind [43, 54]. The more successfully merged requirements are provided, the more security-usability alignment balancing is achieved, which is usable-security. Therefore, the requirements of the two groups, the security requirements group and the usability requirements group, are visited again and analyzed for merging purpose. Once new usable-security requirements are derived from the existing security and usability requirements, they are rated \( MUsa\_SecR \) through averaging the security and usability rates \( (SecR, UsaR) \) of all contributing requirements (CSR and CUR) multiplied by 2 as follows:

\[
MUsa\_SecR_t = \frac{\left( \frac{\sum_i^{CSR} SecR_i + \sum_j^{CUR} UsaR_j}{CSR + CUR} \right) \times 2}{3}
\]  

(3.15)

The overall merged usable-security rate \( MUSA\_SEC \) is calculated from the following equation:

\[
MUSA\_SEC = \sum_{l}^{MUSR} MUsa\_SecR_t
\]  

(3.16)

The overall prediction of usable-security rate \( USA\_SEC \) for the entire system is calculated by the following equation:

\[
USA\_SEC = IUSA\_SEC + MUSA\_SEC.
\]  

(3.17)
3.2.2 Utility Functions

Utility Function (UF) is a relatively straightforward methodology to deal with conflicting objectives and capture stakeholders’ attitude about predictive assessment and evaluation of trade-offs [22]. Utility function is usually used in systems engineering and management for decision and risk analysis purposes. There are various models of utility function. One is the Additive Utility Function (AUF) that is used to estimate total utility of conflicting objectives. Another utility function model is the Individual Utility Function (IUF). The IUF is used to predictively estimate utilities for subjectively measurable/non-measurable objectives. More details about the above utility function models are available on [22]. Usable-security is a subjectively measurable hybrid software quality attribute that is based on two conflicting quality attributes, security and usability, along with consideration to HCI-SEC principles [36, 54]; therefore, the utility function models can be tamable for usable-security assessment during the requirements engineering phase. Assessing usable-security during the requirements phase can provide clear prediction about balancing security and usability early in the software development process. Based on the requirements filtering and merging process, both the individual and additive utility function models can be used to assess usable-security.

First, the ratio-based IUF is used to calculate weights of the software quality attributes: security, usability, and usable-security. The ratios of security \( R_{SEC} \), usability \( R_{USA} \), and usable-security \( R_{USA-SEC} \) are derived as the following calculation, where \( \alpha \) represents an attribute (security, usability, or usable-security) and \( \beta \) represents another attribute: \( \alpha \) is \( \frac{\alpha}{\beta} \) times as important as \( \beta \). For instance, the ratio of security over usability is calculated based on the above calculation as follows: Security is \( \frac{SEC}{USA} \) times as important as usability. Each attribute’s accumulative ratio is then
calculated through summing its ratios over all the attributes. For instance, the security accumulative ratio is calculated by summing the security ratios over all the attributes as follows:

\[ R_{SEC} = \frac{SEC}{SEC} + \frac{SEC}{USA} + \frac{SEC}{USA,SEC} \]  \hspace{1cm} (3.18)

After the accumulative ratios of the attributes are derived, each attribute is weighted based on the following equations, where security, usability, and usable-security weights are \( K_{SEC} \), \( K_{USA} \), and \( K_{USA,SEC} \) respectively [22], \( i \) represents a quality attribute, and \( QA \) represents the number of all the quality attributes:

\[ K_{SEC} = \frac{1}{\sum_{i}^{QA} R_i} * R_{SEC} \]  \hspace{1cm} (3.19)

\[ K_{USA} = \frac{1}{\sum_{i}^{QA} R_i} * R_{USA} \]  \hspace{1cm} (3.20)

\[ K_{USA,SEC} = \frac{1}{\sum_{i}^{QA} R_i} * R_{USA,SEC} \]  \hspace{1cm} (3.21)

Second, the IUF is used to calculate weights of each of security and usability properties based on their ratios \( (R) \) and pointing \( (P) \), where the starting pointing value is five \( (5) \). The following equations are used to calculate the weights of security properties: confidentiality \( (K_c) \), integrity \( (K_n) \), and availability \( (K_a) \) [22], \( i \) represents a property, and \( SP \) represents the number of all properties. The weights calculation is applied as follows:

\[ K_c = \frac{1}{\sum_{i}^{SP} P_i} * P_c \]  \hspace{1cm} (3.22)
The following equations are used to calculate the weights of usability properties: effectiveness ($K_e$), efficiency ($K_f$), and satisfaction ($K_s$) [22], $j$ represents a property, and $UP$ represents the number of all properties. The weights calculation is applied as follows:

$$K_e = \frac{1}{\sum_{j}^{UP} p_j} * P_e$$  \hspace{1cm} (3.25)$$

$$K_f = \frac{1}{\sum_{j}^{UP} p_j} * P_f$$  \hspace{1cm} (3.26)$$

$$K_s = \frac{1}{\sum_{j}^{UP} p_j} * P_s$$  \hspace{1cm} (3.27)$$

Third, the IUF is used to calculate utilities of security and usability properties based on the ratios ($R$), pointing ($P$) where the starting pointing value is five (5), and the following equations are used to find values of constants $a$ and $b$ for security and usability individually [22]:

$$b = \frac{1}{(-1) * min_{properties} P + max_{properties} P}$$  \hspace{1cm} (3.28)$$

$$a = ((-1) * min_{properties} P) * b$$  \hspace{1cm} (3.29)$$
Based on the values of the constants $a$ and $b$ on security, the following equations are used to calculate the utilities ($U$) of security properties: confidentiality ($U_c$), integrity ($U_n$), and availability ($U_a$):

$$U_c = a + (b \times P_c)$$

(3.30)

$$U_n = a + (b \times P_n)$$

(3.31)

$$U_a = a + (b \times P_a)$$

(3.32)

Similarly, based on the values of the constants $a$ and $b$ on usability, the following equations are used to calculate the utilities ($U$) of usability properties: effectiveness ($U_e$), efficiency ($U_f$), and satisfaction ($U_s$) [22]:

$$U_e = a + (b \times P_e)$$

(3.33)

$$U_f = a + (b \times P_f)$$

(3.34)

$$U_s = a + (b \times P_s)$$

(3.35)

Fourth, the AUF is used to calculate the overall utility of the quality attributes based on their properties’ weights and utility values. The following equations represent the AUF for the security and usability quality attributes:

$$U_{SEC}(c, n, a) = K_c \times U_c + K_n \times U_n + K_a \times U_a,$$

(3.36)
Usable-security utility \(U_{USA,SEC}\) differs from the utility of security and usability because usable-security does not have properties. It is a result of merging the two quality attributes: security and usability. Therefore, the following equation is used to calculate the utility of usable-security:

\[
U_{USA,SEC} (SEC, USA) = K_{SEC} * U_{SEC} + K_{USA} * U_{USA} \tag{3.38}
\]

### 3.2.3 Decision Trees

A Decision Tree (DT) is a tool used during the process of modeling decisions [22]. It is a method of structuring different objectives’ elements in order to make decisions for using the objectives based on displaying all of the minute details. Quality attributes in general, and security, usability, and usable-security in particular, are objectives of software development within the scope of our framework. More information about the Decision Tree is available in [22].

To obtain the overall utility value of the Decision Tree for the three quality attributes, the weights and utilities of each attribute are multiplied as the following equations. To obtain the overall utility for usable-security, we subtracted the gap between security and usability utilities as one important factor that plays a role in assessing the usability-security interaction (usable-security):

\[
DTU_{SEC} = K_{SEC} * U_{SEC} \tag{3.39}
\]
Figure 3.2: The Decision Tree for Scenario 1: This Decision Tree displays all of the minute details for assessing the three quality attributes: security, usability, and usable-security of Scenario 1.

\[
DTU_{USA} = K_{USA} \times U_{USA} 
\]

\[
DTU_{USA,SEC} = (K_{USA,SEC} \times U_{USA,SEC}) - |DTU_{SEC} - DTU_{USA}|
\]

The details of calculating the overall utility values of the quality attributes \((DTU_{SEC}, DTU_{USA}, \text{and } DTU_{USA,SEC})\) are displayed on the decision trees of scenarios 1, 2, and 3 that are used to demonstrate the assessment framework (see Figures 3.2, 3.3, and 3.4) respectively.
Figure 3.3: The Decision Tree for Scenario 2: This Decision Tree displays all of the minute details for assessing the three quality attributes: security, usability, and usable-security of Scenario 2.
Figure 3.4: The Decision Tree for Scenario 3: This Decision Tree displays all of the minute details for assessing the three quality attributes: security, usability, and usable-security of Scenario 3.
Finally, to get the final assessment value for the three quality attributes, the resulting Decision Tree utility value of each attribute is divided by the total summing of the three Decision Tree utilities of all the three attributes as the following equations, where the summing of the results must equals 1:

$$ASS_{SEC} = \frac{DTU_{SEC}}{DTU_{SEC} + DTU_{USA} + DTU_{USA,SEC}}$$ \hspace{1cm} (3.42)

$$ASS_{USA} = \frac{DTU_{USA}}{DTU_{SEC} + DTU_{USA} + DTU_{USA,SEC}}$$ \hspace{1cm} (3.43)

$$ASS_{USA,SEC} = \frac{DTU_{USA,SEC}}{DTU_{SEC} + DTU_{USA} + DTU_{USA,SEC}}$$ \hspace{1cm} (3.44)

### 3.3 Results and Discussion

For each of the three scenarios (see Figure 3.5), we first created a baseline by applying the static OWASP RRM [78] and SALUTA APT [40] assessments for security and usability requirements respectively, using predetermined values for rating each of the two attributes’ requirements. Then, we applied the AFUS approach to reassess the balance among the three attributes.

The table below shows the outcomes from the static assessments, before applying the AFUS; and the dynamic assessments, after applying the AFUS. As seen in Table 3.3, the results of all the three scenarios show moderate difference in assessing security and usability attributes before and after applying AFUS. Moreover, usable-security weight was only incorporated after applying AFUS. The AFUS assessments of the first two scenarios show a moderate to small range of assessment gap between the security and usability attributes, 16.96 percent and 10.44 percent respectively. The third scenario
Figure 3.5: The "Before" and "After" Assessment Results
Table 3.3: The Assessment Results for Scenarios 1, 2, and 3

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Assessment</th>
<th>Security</th>
<th>Usable-Security</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>Before applying AFUS</td>
<td>0.364</td>
<td>No Assessment</td>
<td>0.636</td>
</tr>
<tr>
<td></td>
<td>After applying AFUS</td>
<td>0.365</td>
<td>0.197</td>
<td>0.439</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Before applying AFUS</td>
<td>0.432</td>
<td>No Assessment</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>After applying AFUS</td>
<td>0.348</td>
<td>0.338</td>
<td>0.315</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>Before applying AFUS</td>
<td>0.504</td>
<td>No Assessment</td>
<td>0.496</td>
</tr>
<tr>
<td></td>
<td>After applying AFUS</td>
<td>0.251</td>
<td>0.500</td>
<td>0.249</td>
</tr>
</tbody>
</table>

showed very little assessment gap between the security and usability attributes, 0.80 percent; therefore, it provided higher weight for usable-security.

### 3.4 Conclusion

In this chapter we have provided an assessment methodology that assesses the conflicting quality attributes of security and usability, along with assessing their intersecting hybrid attribute of usable-security. Chapter 4 relies on the work of this chapter. It provides a set of usable-security design guidelines. However, each of the two components (Chapters 3 and 4) can be used independently. In Chapter 6 we will explain how Chapters 3 and 4 are linked together through presenting a case study that demonstrates processing the components of the USEF.
Chapter 4

Usable-Security Guidelines during Software Design

Preface

Requirements engineering and design are the first two phases of the Software Development Life-Cycle. Considerable research has addressed the requirements phase and a number of well-regarded tools exist to assist with that process. The design phase can also make use of a wide range of tools, including design principles, activities, best practices, techniques, and patterns, to improve the incorporation of requirements into the software design documents. However, the process of selecting the appropriate design tools to support each requirement is a complex task that requires considerable training and experience. It is also possible that design tools selected for different requirements can conflict with each other, reducing their effectiveness, increasing complexity, impacting usability or potentially causing security vulnerabilities. In this chapter, we propose guidelines for selecting appropriate design tools to support the
integration of usability and security requirements in the software design phase and to resolve conflicts between those tools. We demonstrate this approach with a case study that illustrates the design tool selection and analysis process.

### 4.1 Introduction

The second phase of the Software Development Life-Cycle (SDLC) addresses the design of a system and makes use of a number of tools that aid in interpreting the non-technical textual documents produced during the requirements phase into a technical document that can be used to construct a software product [58]. A wide variety of tools has been developed for use during the design phase, such as design principles, activities, best practices, techniques, and patterns [7]. It is important that decisions made during the design phase address specific requirements and this often requires feedback from the design phase to the requirements engineering process.

Unfortunately, security and usability are often regarded as non-functional quality attributes. As a result, they may not be considered until near the end of the development process, or when negative security or usability issues appear in software products after deployment. There is a body of research on the inclusion of security best practices and patterns during the design phase [58, 75, 97, 48]. Similarly, usability techniques and patterns for the design phase have received some attention [33, 34, 39] and research has also shown the benefits of integrating security and usability into usable-security to create a balance between those two attributes that enhances both. Additionally, design tools for usable-security have been introduced [79, 43].

The process of linking the available design tools to security and usability requirements needs further research to ensure that those attributes are properly
integrated into the software development process. Design tools (such as patterns) are not usually identified and selected during the requirements phase [7]. Delaying those choices until the beginning of the architectural design process places constraints on the design of the system [39]. Therefore, it is usually better to make those decisions as early as possible in the design phase.

The work proposed here provides systematic guidance for linking the usability, security, and usable-security requirements to appropriate design tools (i.e. principles, practices, patterns, and techniques) that can be used during the architectural design stage. This chapter is structured as follows: Section 4.2 introduces the proposed guidance; Section 4.3 presents a case study to illustrate and demonstrate the guideline in details; and finally, Section 4.4 concludes this chapter and describes future directions for developing the proposed guidelines.

4.2 Guidelines for Usable-Security

Figure 1 shows the proposed process for selecting appropriate design tools to support the requirements specified in the first phase of the SDLC. There are eight steps in this process: 1) selection of priority sensitivity; 2) requirements filtering; 3) selection of appropriate design tools; 4) creation of a linkage mapping table; 5) generation of new design tools; 6) analysis of the trade-off matrix; 7) a check on how well requirements and design tools match; and 8) incorporation of the results (if step 7 is successful) or feedback to the requirements phase (if step 7 shows that the requirements should be modified).
4.2.1 Selection of Priority Sensitivity

The first step is to determine an important goal of the design phase by clearly stating the targeted priority levels for security, usability, and usable-security. This step is directly related to the priority levels assigned to requirements in the previous phase. There are two different strategies which could be followed: either identifying a unified priority target that applies to all the requirements (generalization), or identifying separate target priority levels for each requirement individually (specialization). In both cases, designers must determine whether they intend to provide a high, medium or low level of sensitivity for the priority of the stated requirements when assigning the design tools. These sensitivity levels are described below.
Higher Sensitivity to Requirement Priorities.

At the higher level of sensitivity, all of the requirements are associated with appropriate design tools (e.g., principles, practices, patterns, or techniques), regardless of the priority assigned in the requirements phase. If any requirement cannot be associated with an appropriate design tool, it cannot be processed through the design phase until this issue is addressed. This will be reported to the requirements engineers as it may require adjustments to the requirements process.

Medium Sensitivity to Requirement Priorities.

The medium level of sensitivity ensures that appropriate design tools are associated with all of the medium to high priority requirements. If any of these requirements cannot be associated with an appropriate design tool, they cannot be processed through the design phase until this issue is addressed. This will also be reported to the requirements engineers. Low priority requirements may optionally be associated with design tools, but a failure to associate any of the low priority requirements will not require corrective action.

Lower Sensitivity to Requirement Priorities.

The lower level of sensitivity only expects that design tools are associated with the high priority requirements. Specifying tools for medium or low priority requirements is optional. To achieve this goal, all high priority requirements must be assigned to appropriate design tools. If any of the high priority requirements cannot be associated with an appropriate design tool, they cannot be processed through the design phase until this issue is addressed and this failure will be reported to the requirements engineers.
4.2.2 Requirements Filtering

Based on the target levels of sensitivity specified in the previous step, the requirements are filtered to determine which of them must be associated with appropriate design tools. It should be emphasized that all requirements will be included in the design phase. This filtering process is intended to separate the requirements that must be associated with design tools from those requirements for which design tools are optional. The outcome of the filtering is used to ensure that requirements with the specified level of priority are associated with appropriate design tools.

4.2.3 Selection of Appropriate Design Tools

As mentioned above, there are a number of tools available for enhancing the software development process, a number of which are particularly useful during the design phase. The most common tools are: design principles, best practices, techniques, activities, and patterns. Among them, patterns are possibly the most widely used because they provide reusable, pre-determined solutions for well-known problems that occur frequently during and after software development [68]. However, suitable patterns have not been developed to address all of the problems that can arise in software development. Therefore, incorporating best practices, techniques, and activities during the design phase helps to fill the gap caused by the lack of patterns for some important tasks [7]. Unfortunately, the utilization of these design tools and techniques varies among software developers and not all developers are able to use the available tools properly [7]. Thus, guidance in the selection of suitable design tools is important to supplement those developers’ skills and experience.
4.2.4 Creation of a Linkage Mapping Table

After specifying the desired requirement sensitivity levels, filtering the requirements based on the specified sensitivity levels, and selecting the suitable types of design tools to be used, the design tools are assigned to the appropriate requirements, where the relationship between the requirements and the design tools in a many-to-many mapping. The mapping idea comes from the SALUTA APT [40], where scenarios and usability are linked in an Attribute Preference Table (APT). We adapted the APT to create a Linkage Mapping Table (LMT) to link requirements, attribute properties, and design tools. Table 4.1 shows an example of the linkage mapping table, where Req represents a requirement, Pro represents attribute properties (i.e., Low, Medium or High priority), and DT represents a specific design tool. Table 4.1 shows the mapping between selected requirements and the design tools that will aid in meeting those requirements during the design phase.

4.2.5 Generation of New Design Tools

It is possible that no design tools are available to support a particular requirement that is entered into the LMT. The diagram in Figure 4.1 shows a solution to this problem, the (optional) Design Tools Generation loop. This option provides an opportunity to generate a new design tool, such as new patterns, design activities, techniques, or
practices, for use by the Design Tools Selection component. This step may also cause a refinement to the linkage mapping table.

### 4.2.6 Analysis of the Trade-Off Matrix

Working through quality attributes that have conflicting objectives (i.e., security and usability) to make them synergistically integrated is difficult [16]. However, dealing with such cases is increasingly common in software development projects. Unfortunately, many of the existing design tools were developed with a narrow focus and conflicts between two or more tools are possible. During this step we address potential conflicts by creating a two-dimensional trade-off matrix that lists all the design tools in the linkage mapping table to allow us to identify possible conflicts between these tools. Table 4.2 depicts the trade-off matrix which is used to refine the linkage mapping table (see Figure 4.1).

### 4.2.7 Checking the Match between Requirements and Design Tools

After completion of the trade-off matrix and refining the linkage mapping table, the entries in the linkage mapping table are matched against the requirements to check the compliance of the design phase with the requirements phase. This step reports any missing design tools or conflicts that indicate an inability to meet the requirements.
during the design phase. The results of this analysis are delivered during the final step to determine whether the design process is complete or needs further work.

4.2.8 Incorporation of Results and Feedback

The results from the previous step are analysed for completeness and consistency. If it is determined that the design phase is complete, the linkage mapping table is passed to the designers and architecture engineers to provide them with the essential design tools for the software product. If not, the results of the analysis are passed back to the requirements engineers for reassessment of the requirements.

4.3 An Example: Secure and Usable Authentication

To better illustrate the advantages of our guidelines, we will use an authentication system that we are currently developing as part of our research in usable-security. Authentication is one of two areas receiving significant attention from the usable-security community [79]. The system under development is a demonstration of an approach that we call the Choice-Based Authentication Approach (CBAA). This approach is based on two concepts. The first is to allow end users to select a preferred authentication method to enhance usability and universal access during the sign-up process [53]. The second concept is to increase the complexity for adversaries by displaying all of the possible authentication methods during the login process. Our aim is to demonstrate the usefulness of the guidelines that we proposed in this chapter. Thus, a brief subset of the CBAA demonstration is presented here for each step of the guideline.
Table 4.3: Security Requirements Filtering

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>CONF</th>
<th>INT</th>
<th>AVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SReq1</td>
<td>The system shall provide one unique login interface.</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SReq2</td>
<td>The system shall show all authentication methods on the login interface.</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>SReq3</td>
<td>The system shall meet the NIST 800-Series.</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SReq4</td>
<td>The system shall not store users’ private information.</td>
<td>6</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>SReq5</td>
<td>The system shall provide a separate sign-up interface.</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

4.3.1 Selection of Priority Sensitivity

As described earlier, this step relies on the requirements engineering phase of the SDLC. In order to identify the target priority levels, we used the results of rating the requirements for the authentication system under development based on the Assessment Framework for Usable-Security (AFUS) [51]. Chapter 3 presents the AFUS in details for usability. Tables 4.3 and 4.4 are used to illustrate the selection of priority sensitivity, where all rating values are selected for higher sensitivity, the rating values starting from six are selected for medium sensitivity, and the rating values starting from seven are selected for the lower sensitivity level. We adopted the generalization strategy that applies one design goal over all the requirements and we used the medium level of priority sensitivity, which means that priority values starting from six should be associated with appropriate design tools. These parameters were specifically to present a typical demonstration of the guidelines.

AFUS is an assessment methodology that we developed based on well-known tools from the Decision Science field. AFUS rates the requirements via adapting two well-known assessment methodologies: the OWASP RRM [78] for security, and the SALUTA APT [40]
4.3.2 Requirements Filtering

Based on the sensitivity targets from the previous step (i.e., “medium” sensitivity), the requirements are filtered to identify any that are not expected to be associated with a design tool. Table 4.3 shows a list of security requirements, where CONF, INT, and AVA represent the security properties: confidentiality, integrity, and availability. At least one of those three properties must meet the expected priority level to be assigned a design tool. For example, in Table 4.3, SReq will be eliminated because it does not have rating values of six or higher and is considered to be a low priority requirement. Similarly, Table 4.4 shows a list of usability requirements, where EFF1, EFF2, and SAT represent the usability properties: effectiveness, efficiency, and satisfaction, respectively.

4.3.3 Selection of Appropriate Design Tools

In our example, all of the available design tools are considered. Therefore, best practices, techniques, activities, and patterns can all be assigned to requirements during the design phase.

4.3.4 Creation of a Linkage Mapping Table

After completion of the above steps of the guideline, a linkage mapping table can be created through associating each requirement with appropriate design tools. For example, the first security requirement SReq on Table 4.5 mandates providing a unified individual login interface that all users should use during the login process. This requirement can be met through applying two well-known security patterns: SINGLE ACCESS POINT [108] and FRONT DOOR [39]. In the same way, usability
Table 4.4: Usability Requirements Filtering

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>EFF1</th>
<th>EFF2</th>
<th>SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UReq_1$</td>
<td>The system shall help the users to avoid making mistakes by providing clear steps to be followed.</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$UReq_2$</td>
<td>The system shall make the users satisfied and want to use it by adopting user-friendly interfaces.</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>$UReq_3$</td>
<td>The system shall allow users to complete the required tasks in an acceptable amount of time.</td>
<td>3</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>$UReq_4$</td>
<td>The system shall not increase the users’ cognitive load through providing consistent steps and interfaces.</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_5$</td>
<td>The system shall provide an acceptable login time that does not exceed the standard login time.</td>
<td>1</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_6$</td>
<td>The system shall not confuse the users by avoiding duplication, double vision, implicit negatives, dangling alternatives.</td>
<td>9</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>
requirements are associated with appropriate design tools as shown in Table 4.6. We provided examples of the design tools used to meet the high priority requirements of Tables 4.3 and 4.4:

\[ DT_1: \text{SINGLE ACCESS POINT} \text{ [108]} \]
\[ DT_2: \text{FULL VIEW WITH ERROR} \text{ [108]} \]
\[ DT_3: \text{SECURITY POLICY ENFORCEMENT} \text{ [48]} \]
\[ DT_4: \text{ANONYMITY SET} \text{ [48]} \]
\[ DT_5: \text{FORM/FIELD VALIDATION} \text{ [39]} \]
\[ DT_6: \text{PROGRESS INDICATION} \text{ [39]} \]
\[ DT_7: \text{MULTI-TASKING} \text{ [39]} \]
\[ DT_8: \text{EMULATION} \text{ [39]} \]
\[ DT_9: \text{ALERTS} \text{ [39]} \]
\[ DT_{10}: \text{SHORTCUTS} \text{ [39]} \]
\[ DT_{11}: \text{COGNITIVE WALK-THROUGH} \text{ [33]} \]
\[ DT_{12}: \text{ACTIONS FOR MULTIPLE OBJECTS} \text{ [39]} \]
\[ DT_{13}: \text{HISTORY LOGGING} \text{ [39]} \]
\[ DT_{14}: \text{PROTOTYPING} \text{ [33]} \]
\[ DT_{15}: \text{WIZARD} \text{ [39]} \]

### 4.3.5 Generation of New Design Tools

As stated in Section 4.2.5, this step is optional and added in order to increase flexibility in the guideline to allow design tools development and generation. However, there are
Table 4.5: The Linkage Mapping Table for the Security Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Rate</th>
<th>Design Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>SReq1</td>
<td>Low</td>
<td>DT1</td>
</tr>
<tr>
<td>SReq2</td>
<td>High</td>
<td>DT2</td>
</tr>
<tr>
<td>SReq3</td>
<td>High</td>
<td>DT3</td>
</tr>
<tr>
<td>SReq4</td>
<td>High</td>
<td>DT4</td>
</tr>
</tbody>
</table>

enough design tools that meet the requirements of our authentication system; therefore, there is no need to generate new tools in this case.

4.3.6 Analysis of the Trade-Off Matrix

The trade-off matrix analysis is used to identify any conflicts between the design tools selected to address the requirements’ needs during the design phase. Table 6.24 shows examples of conflicting cases where the adopted design tools work against one another. For instance, the FULL VIEW WITH ERROR pattern [108] conflicts with the usability activity for avoiding dangling alternatives and double vision that is ACTIONS FOR MULTIPLE OBJECTS [39]. Another conflicting case appears when the SECURITY POLICY ENFORCEMENT pattern [48] is used to ensure applying security policy, where it may conflict with the Cognitive Walk-through [33] usability technique. These conflicting design tools have been identified and analyzed to allow us to address the conflicts. In the case of our authentication, the conflict between design tools is resolved by adopting usable-security design tools such as Interaction Design activities [33].
Table 4.6: The Linkage Mapping Table for the Usability Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>EFF1</th>
<th>EFF2</th>
<th>SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>UReq1</td>
<td>Rate</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT5</td>
<td></td>
</tr>
<tr>
<td>UReq2</td>
<td>Rate</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT6</td>
<td></td>
</tr>
<tr>
<td>UReq3</td>
<td>Rate</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT7</td>
<td>DT8</td>
</tr>
<tr>
<td>UReq4</td>
<td>Rate</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT11</td>
<td>DT9</td>
</tr>
<tr>
<td>UReq5</td>
<td>Rate</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT13</td>
<td>DT14</td>
</tr>
<tr>
<td>UReq6</td>
<td>Rate</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>Design Tool</td>
<td>DT12</td>
<td>DT15</td>
</tr>
</tbody>
</table>

Table 4.7: Design Tools Trade-Off Matrix for the CBAA Demonstration

<table>
<thead>
<tr>
<th>DT1</th>
<th>DT12</th>
<th>DT3</th>
<th>DT11</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT12</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT3</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DT11</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

and LEAST SURPRISE / LEAST ASTONISHMENT and PROVIDE STANDARDIZED SECURITY POLICIES [43].

4.3.7 Checking the Match between Requirements and Design Tools

After refining and completing the linkage mapping tables, a matching check is performed to ensure that the requirements have been completely covered by appropriate design tools and there are no missing tools, as missing tools indicate that the design phase is limited and therefore unable to properly meet the needs of all of the high priority requirements. In our demonstration, we could successfully cover all the requirements
through assigning each requirement with the appropriate design tools. Conflicts have been resolved by applying usable-security principles and patterns [43].

4.3.8 Incorporation of Results and Feedback

After a successful analysis, the completed linkage mapping tables are ready to be passed to the architecture engineers so they can use the tables to select the appropriate design tools during the architecture design activities.

4.4 Conclusion

In this chapter, we proposed guidelines to improve the selection of the appropriate design tools for security, usability, and usable-security to ensure that the design process meets the requirements’ needs. Moreover, our guidelines help to identify trade-offs in the selection of design tools by discovering conflicts and helping to overcome contradictions in design decisions. We recognize that more work is needed to determine how well our approach enhances the development of software products in terms of both linking the requirements to the design tools, and integrating security and usability synergistically for better usable-security.
Chapter 5

Usable-Security Measurement during Evaluation

Preface

Developing software products which align security and usability to make a synergistic relationship between security and usability is an engineering process that starts from the first phase of the Software Development Life-Cycle (SDLC), and continues through the rest of the phases: design, construction, and testing. A summative evaluation of such a process must be done after the software product is completely developed with careful attention to measuring the alignment between security and usability (i.e.: usable-security), and integrating such alignment properly within the SDLC. This chapter proposes a usable-security measuring matrix that provides a summative evaluation of the whole process of applying usable-security on software products.
5.1 Introduction

As a result of the increased adoption of a user-centered approach to improve human-computer interaction and increasing concerns regarding computer-related privacy and security issues, researchers and scientists have proposed many design techniques that incorporate usable-security [32, 43, 54, 25]. Those techniques are used to build software systems that seek to balance usable and secure. However, if we take a careful look at the currently available software systems, we often find that quality attributes, such as security and usability, are applied and evaluated separately [8, 40, 78].

Therefore, the vast majority of those systems are built and deployed without giving proper attention to evaluating the state of integrating the quality attributes, such as usable-security. It is important to ensure that software systems are usable and secure [74, 54], so that people can rely on such systems and interact with them as if interacting with actual human beings. People ought to be able to use and deal with systems with hands-on ease as this helps in a better cooperation between the people and the systems [33, 40, 25]. Also, people must be able to trust that the systems are properly secure against any malicious penetrative behavior that may cause intentional undesired changes or updates during the human-computer interaction [7, 97].

Unfortunately, without careful consideration of the balance between usability and security, a system can be designed so that those properties will actually work against each other; in such cases they cannot be aligned and evaluated together as one jointly concept [1, 67, 104]. However, there are some studies that try to bridge usability and security to help having better performance and quality for computer systems [29]. We adopted the approach of bridging usability and security attributes together as one combined attribute: usable-security. One way to achieve this goal is using models
and frameworks that help in assessing software usability, security, and usable-security requirements based on formative evaluation approach [40, 78, 51]. Assessing usable-security helps in predicting to which level software systems could be simultaneously usable and secure [51]. This chapter proposes a summative usable-security evaluation matrix based on the usable-security assessment models and frameworks.

Section 5.2 introduces a proposed mathematical-based evaluation model to help determining the levels of usability and security quality attributes. Section 5.3 introduces a proposed matrix to evaluate usable-security of software systems. Finally, Section 5.5 concludes this chapter.

5.2 Usability-Security Measuring and Evaluation

In order to develop a useful evaluation methodology, developers should have appropriate measuring techniques before the evaluation process begins. Access to suitable metrics gives developers the ability to predict the evaluation’s outcome. Usable-security is difficult to measure; consequently, it is difficult to evaluate as well [26, 43].

This work proposes new measuring and evaluation techniques based on OWASP [78] and SAULTA [40]. The evaluation technique serves as extended research work for Chapter 3, that is: the Assessment Framework for Usable-Security (AFUS) [51]. The following sections describe methods for measuring and evaluating usability and security.

5.2.1 Usability Evaluation

The term usability is defined by the International Standard Organization (ISO) as the range in which a product can be operated by legitimate users to satisfactorily perform specific tasks in an effective, efficient, and specified way [44]. Some others add
accuracy, memorability, and learnability as secondary usability factors [67]. This paper focuses only on the three factors of the ISO definition.

In order to evaluate the targeted usability level on any developed software system during the evaluation phase, the application state (level) of each of the usability properties (efficiency, effectiveness, and satisfaction) must be focused on, as well as providing a method to determine a measurable way to evaluate these properties on software systems. The following evaluation equations are adapted from Bevan and Macleod’s concepts [69, 13].

**Effectiveness**

Software systems can only be considered as effective if their users are able to achieve their goal of operating such systems. The effectiveness property can be measured based on a goal-centered view by counting the number of successful tasks that legitimate users perform [49], including actions like, when software systems allow users to successfully create their passwords, login using their previously created passwords, or provide their biometric traits. Equation 5.1 can be used to evaluate the effectiveness for software systems, where $n$ represents the total number of accepted tasks that legitimate users perform, $R$ represents the result of each performed task’s trial (either ”failure” or ”success”), $EF1$ represents the system effectiveness rate.

$$EF1 = \frac{1}{n} \sum_{i=1}^{n} \delta(R[i])$$

Where \( \delta(\theta) \) is defined as: 

\[
\begin{align*}
\delta(\theta) &= \begin{cases} 
0 & \text{if } \theta = \text{failure} \\
1 & \text{if } \theta = \text{success}
\end{cases}
\end{align*}
\]
If the result of system effectiveness rate, $EF_1 \geq 0.9$, this means that the software system is highly effective. On the other hand, if the result of $EF_1 < 0.3$, this means that the system is ineffective and needs further enhancement. The following illustrates the system effectiveness rate measurement:

$$\text{Effectiveness Measurement} = \begin{cases} 
    \text{Highly effective} & \text{if } EF_1 \geq 0.9 \\
    \text{Mostly effective} & \text{if } 0.7 \leq EF_1 < 0.9 \\
    \text{Effective} & \text{if } 0.6 \leq EF_1 < 0.7 \\
    \text{Some effective} & \text{if } 0.3 \leq EF_1 < 0.6 \\
    \text{Ineffective} & \text{if } EF_1 < 0.3 
\end{cases}$$

The above effectiveness measurement reveals that at least 90 percent of the accepted task trials must succeed in order for the software system to be considered highly effective. After measuring the effectiveness, it is evaluated based on the following numerical evaluation, because this provides the consistency among all of the usability properties. The effectiveness evaluation is as follows:

$$\text{Effectiveness Evaluation} = \begin{cases} 
    EF_1 = 0.9 & \text{if highly effective} \\
    EF_1 = 0.7 & \text{if mostly effective} \\
    EF_1 = 0.6 & \text{if effective} \\
    EF_1 = 0.3 & \text{if some effective} \\
    EF_1 = 0.1 & \text{if ineffective} 
\end{cases}$$

**Efficiency**

Efficient software systems must complete a specific task or process to reach a particular goal within an acceptable amount of time. In contrast, any system is not efficient if
it takes an unacceptable amount of time to achieve a specific goal. The efficiency property is important because both the vendors and the users will not rely on a system that takes a long time to perform a specific task (for instance: authentication). The measurement used to evaluate the efficiency is the amount of time that is consumed in order to achieve a particular goal or to complete a particular task. Equation 5.2 depicts the evaluation, where \( n \) represents number of trials to perform a particular task, \( \beta \) represents the standard average amount of time to finish such task, and \( T \) represents the amount of time consumed to perform the task on each trial.

\[
EF^2 = \frac{\frac{1}{n} \sum_{i=1}^{n} T[i]}{\beta}
\]  

If the result of system efficiency rate, \( EF^2 \), is less than 1, this means that the software system is highly efficient. In contrast, if the resultant value of \( EF^2 \) is greater than or equal to 1.50, this means that the software system is inefficient because the average amount of time consumed to perform that task is away beneath the standard average, \( \beta \).

The following illustrates the system efficiency rate measurement:

\[
\text{Efficiency Measurement} = \begin{cases} 
\text{Highly efficient} & \text{if } EF^2 < 1.0 \\
\text{Mostly efficient} & \text{if } EF^2 = 1.0 \\
\text{Efficient} & \text{if } 1.0 < EF^2 \leq 1.2 \\
\text{Some efficient} & \text{if } 1.2 < EF^2 < 1.5 \\
\text{Inefficient} & \text{if } EF^2 \geq 1.5 
\end{cases}
\]

The above system efficiency rate equation, \( EF^2 \), shows that the average time of set of trials to perform a particular task must be less than or equal to 1.2 of the standard average amount of time consumed to perform such task in order for a software system...
to be considered efficient. After measuring the efficiency, it is evaluated based on the following numerical evaluation because this provides the consistency among all of the usability properties. The efficiency evaluation is as follows:

\[
\text{Efficiency Evaluation} = \begin{cases} 
EF2 = 0.9 & \text{if highly efficient} \\
EF2 = 0.7 & \text{if mostly efficient} \\
EF2 = 0.6 & \text{if efficient} \\
EF2 = 0.3 & \text{if some efficient} \\
EF2 = 0.1 & \text{if inefficient}
\end{cases}
\]

**Satisfaction**

For a software system to be satisfactory, both of the vendors and the users must be happy with the system. This is determined by the willingness of both vendors and users to rely on and reuse the system, respectively. Satisfaction is strongly affected by the vendors’ and the users’ mood [67]. It is most likely to be perceived as satisfactory if it is both simultaneously effective and efficient. Despite the previous fact, some vendors or users might consider a system as unsatisfactory even if it is both effective and efficient; therefore it is desirable to evaluate satisfaction levels in software systems. Evaluating the satisfaction property can prove to be a challenge due to the difficulty of having accurate measurement tools. The best way to evaluate satisfaction is through vendor/user questionnaires such as in SUMI [69] and the SUS [101] surveys, or interviews [67]. Satisfaction is evaluated through involving HCI-SEC principles that are related to user-centered approach [46] that focus on the ease of use, the degree of happiness, and the degree of confusion. Based on the results of the given survey, user satisfaction is
measured as follows:

\[
\text{Satisfaction Measurement} = \begin{cases} 
\text{Highly satisfied} & \text{if } SA1 \geq 0.9 \\
\text{Mostly satisfied} & \text{if } 0.7 \leq SA1 < 0.9 \\
\text{Satisfied} & \text{if } 0.6 \leq SA1 < 0.7 \\
\text{Some satisfied} & \text{if } 0.3 \leq SA1 < 0.6 \\
\text{Not satisfied} & \text{if } SA1 < 0.3 
\end{cases}
\]

After measuring user satisfaction based on the above degree of satisfaction via surveys, it is evaluated based on the following numerical evaluation because this provides the consistency among all of the usability properties. The satisfaction evaluation is as follows:

\[
\text{Satisfaction Evaluation} = \begin{cases} 
SA1 = 0.9 & \text{if highly satisfied} \\
SA1 = 0.7 & \text{if mostly satisfied} \\
SA1 = 0.6 & \text{if satisfied} \\
SA1 = 0.3 & \text{if some satisfied} \\
SA1 = 0.1 & \text{if not satisfied} 
\end{cases}
\]

Evaluating the above standard usability properties leads to an overall usability evaluation through summing the evaluations of the three properties (effectiveness, efficiency, and user satisfaction). Equation 5.3 illustrates the calculation, where \( \alpha \) represents the usability:

\[
\alpha = \frac{EF1 + EF2 + SA1}{3} \quad (5.3)
\]
Equation 5.9 provides one of the following usability ($\alpha$) statuses:

$$\text{Usability Measurement} = \begin{cases} 
\text{Highly usable} & \text{if } \alpha \geq 0.9 \\
\text{Mostly usable} & \text{if } 0.7 \leq \alpha < 0.9 \\
\text{Usable} & \text{if } 0.6 \leq \alpha < 0.7 \\
\text{Some usable} & \text{if } 0.3 \leq \alpha < 0.6 \\
\text{Not usable} & \text{if } \alpha < 0.3 
\end{cases}$$

Like evaluating usability properties, overall usability of software systems is evaluated based on the same numerical evaluation methodology. This provides the evaluation consistency among three quality attributes under investigation (usability, security, and usable-security). Usability evaluation is as follows:

$$\text{Usability Evaluation} = \begin{cases} 
\alpha = 0.9 & \text{if highly usable} \\
\alpha = 0.7 & \text{if mostly usable} \\
\alpha = 0.6 & \text{if usable} \\
\alpha = 0.3 & \text{if some usable} \\
\alpha = 0.1 & \text{if not usable} 
\end{cases}$$

### 5.2.2 Security Evaluation

The term "security" is defined in various ways. Essentially, system security is a set of methods and techniques used to prevent weaknesses from being exploited by applying three security goals: confidentiality, integrity, and availability [80]. Software systems, including security and secure systems, have vulnerabilities that need to be discovered and closed, or at least protected from being imposed. Therefore, in order for
a system to be within an acceptable level of security, the three security goals / properties (confidentiality, integrity, and availability) must be applied and achieved. The following is an analysis on the importance of each of the security goals to software systems, how those properties are applied to the systems, and how they are measurably evaluated.

**Confidentiality**

Confidentiality is a factor or goal of any secure system. Confidentiality is identified as the ability to grant access only to authorized users. In order for a software system to be considered confidential, it must limit access to its data, features, and services solely to authorized users. If unauthorized users gain access to computer systems, confidential information may be accessed and then used against the systems’ vendors and users. To measure software systems’ confidentiality, Equation 5.4 is applied where \( n \) represents the total number of access trials, \( \alpha \) represents an individual access trial (either “false access” or “true access”), \( TAR \) represents true access rate, \( FAR \) represents false access rate, and \( CO1 \) represents system confidentiality evaluation value.

\[
CO1 = TAR \tag{5.4}
\]

\[
TAR = \frac{\sum_{i=1}^{n} \delta(\alpha[i])}{n} \tag{5.5}
\]

Where \( \delta(\theta) \) is defined as: \( \delta(\theta) = \begin{cases} 1 & \text{if } \theta = \text{true access} \\ 0 & \text{if } \theta = \text{false access} \end{cases} \)

and

\[
FAR = 1 - TAR \tag{5.6}
\]
If the result of system confidentiality rate, \( CO1 \), is greater than or equal to 0.99, the system is highly confidential. In contrast, if the resulted value of \( CO1 \) is less than or equal to 0.80, the system ailed to provide confidentiality because access was granted for too many unauthorized users. The following illustrates the confidentiality rate measurement:

\[
\text{Confidentiality Measurement} = \begin{cases} 
\text{Highly confidential} & \text{if } CO1 \geq 0.99 \\
\text{Mostly confidential} & \text{if } 0.95 \leq CO1 < 0.99 \\
\text{Confidential} & \text{if } 0.90 \leq CO1 < 0.95 \\
\text{Some confidential} & \text{if } 0.80 \leq CO1 < 0.90 \\
\text{Non-confidential} & \text{if } CO1 < 0.80 
\end{cases}
\]

As a result, in order for any software system to be confidential, its confidentiality, \( CO1 \), must be greater than or equal to 0.90, which means that at least 90 percent of the unauthorized trials are not granted access to the system. At the same time, all of the authorized trials are granted access. After the measurement process, confidentiality is evaluated based on the following numerical evaluation because this provides the consistency among all of the security properties. The confidentiality evaluation is as follows:

\[
\text{Confidentiality Evaluation} = \begin{cases} 
CO1 = 0.9 & \text{if highly confidential} \\
CO1 = 0.7 & \text{if mostly confidential} \\
CO1 = 0.6 & \text{if confidential} \\
CO1 = 0.3 & \text{if some confidential} \\
CO1 = 0.1 & \text{if non-confidential} 
\end{cases}
\]
Integrity

Integrity means that for the authorized users, the system does not allow them to perform tasks in an improper way, and protects the data from any unauthorized alteration. Integrity is a crucial factor for any system that seeks true security. As having usable-security evaluation for software systems is the goal of this chapter, the integrity property must be applied and adapted in order to make such systems secure. This property is achieved by enabling systems to create auto-backup and auto-check using proper techniques and tools like hashing, which is the process of comparing backup files with the same files on the system. Equation 5.7 depicts the integrity calculation on software systems, where \( n \) represents the total number of selected files for hashing, \( \text{IN1} \) represents the integrity evaluation result.

\[
\text{IN1} = \frac{1}{n} \sum_{i=1}^{n} \delta \left( \frac{\text{systemfile}[i]}{\text{backupfile}[i]} \right)
\]  (5.7)

Where \( \delta(\theta) \) is defined as:

\[
\delta(\theta) = \begin{cases} 
1 & \text{if } \theta = 1 \\
0 & \text{if } \theta \neq 1
\end{cases}
\]

If the result of system integrity, \( \text{IN1} \), is greater than or equals to 0.99, the system is highly integral because the result of the hashing (comparing between the system and the backup files) indicates that there is no any unauthorized alteration on the system file. In contrast, if the resulted value of \( \text{IN1} \) is less than 0.80, this means that the system is not integral because the occurred alteration is considered as critical. The following illustrates the system integrity measurement:
Integrity Measurement =

\[
\begin{cases}
\text{Highly integral} & \text{if } INI \geq 0.99 \\
\text{Mostly integral} & \text{if } 0.95 \leq INI < 0.99 \\
\text{Integral} & \text{if } 0.90 \leq INI < 0.95 \\
\text{Some integral} & \text{if } 0.80 \leq INI < 0.90 \\
\text{Non- integral} & \text{if } INI < 0.80
\end{cases}
\]

As a result, in order for any software system to be integral, its integrity, $INI$ must be at least equals to 0.90, indicating that the system’s files are not critically altered in an unauthorized process. After the measurement process, integrity is evaluated based on the following numerical evaluation because this provides the consistency among all of the security properties. The integrity evaluation is as follows:

\[
\begin{cases}
INI = 0.9 & \text{if highly integral} \\
INI = 0.7 & \text{if mostly integral} \\
INI = 0.6 & \text{if integral} \\
INI = 0.3 & \text{if some integral} \\
INI = 0.1 & \text{if non- integral}
\end{cases}
\]

**Availability**

Availability is a security factor where the system’s services, contents, or data, must be available at any time an authorized user needs to access them. This property is important for all software systems, since for a system to reach an acceptable level of security, it has to prevent all unavailability incidents. Availability is measured based upon the number of success services or data access requests a system receives. If all of the requests produce a response at any time, it means that the system is highly available.
Equation 5.8 shows the availability calculation, where \( n \) represents the total number of access trials, \( \alpha \) represents an individual access trial (either “available” or ”unavailable”), and \( AVI \) represents availability rate.

\[
AVI = \frac{1}{n} \sum_{i=1}^{n} \delta(\alpha[i])
\]  

(5.8)

Where \( \delta(\theta) \) is defined as:

\[
\delta(\theta) = \begin{cases} 
1 & \text{if } \theta \text{ is available} \\
0 & \text{if } \theta \text{ is unavailable}
\end{cases}
\]

If the result of system availability evaluation, \( AVI \), is greater than or equal to 0.99, the software system is highly available because it is accessible to its authorized users whenever needed. In contrast, if the resulted value of \( AVI \) is less than 0.80, this means that the system is unavailable because there is more than 20 percent unavailability incident occurred. The following illustrates the availability measurement:

\[
\text{Availability Measurement} = \begin{cases} 
\text{Highly available} & \text{if } AVI \geq 0.99 \\
\text{Mostly available} & \text{if } 0.95 \leq AVI < 0.99 \\
\text{Available} & \text{if } 0.90 \leq AVI < 0.95 \\
\text{Some available} & \text{if } 0.80 \leq AVI < 0.90 \\
\text{Unavailable} & \text{if } AVI < 0.80
\end{cases}
\]

As a result, in order for any system to be available, the availability, \( AVI \), must be greater than or equal to 0.90, which means that at least 90 percent of the system access trials are served and received suitable responses. After the measurement process, availability is evaluated based on the following numerical evaluation, because this provides the consistency among all of the security properties. The availability evaluation
Availability Evaluation is as follows:

\[
\text{Availability Evaluation} = \begin{cases} 
AVI = 0.9 & \text{if highly available} \\
AVI = 0.7 & \text{if mostly available} \\
AVI = 0.6 & \text{if available} \\
AVI = 0.3 & \text{if some available} \\
AVI = 0.1 & \text{if unavailable}
\end{cases}
\]

Evaluating the above standard security properties leads to an overall security evaluation through summing the evaluations of the three properties (confidentiality, integrity, and availability) divided by three. Equation 5.9 illustrates the calculation, where \( \beta \) represents the security:

\[
\beta = \frac{COI + IN1 + AVI}{3} \tag{5.9}
\]

Equation 5.9 provides one of the following security \( (\beta) \) statuses:

\[
\text{Security Measurement} = \begin{cases} 
\text{Highly secure} & \text{if } \beta \geq 0.9 \\
\text{Mostly secure} & \text{if } 0.7 \leq \beta < 0.9 \\
\text{Secure} & \text{if } 0.6 \leq \beta < 0.7 \\
\text{Some secure} & \text{if } 0.3 \leq \beta < 0.6 \\
\text{Not secure} & \text{if } \beta < 0.3
\end{cases}
\]

Same as evaluating security properties, overall security of software systems is evaluated based on the same numerical evaluation methodology, because this provides the evaluation consistency among three quality attributes under investigation. Security
evaluation is as follows:

\[
\beta = \begin{cases} 
0.9 & \text{if highly secure} \\
0.7 & \text{if mostly secure} \\
0.6 & \text{if secure} \\
0.3 & \text{if some secure} \\
0.1 & \text{if not secure}
\end{cases}
\]

### 5.3 Usable-Security Evaluation Matrix

Analysis and measurement studies of usability and security properties on systems produced detailed understanding of the nature and structure of systems’ usable-security. Based on the previous section, a usable-security evaluation matrix can be constructed that can be used as a guide to achieve the overall goal of this paper, which is evaluation and enhancement for software systems to be both more usable and secure (see Figure 5.1).

The matrix shown in Figure 5.1 is a proposed method for evaluating usable-security of software systems. Systems’ usable-security can be evaluated by using Equations 5.3 and 5.9 that evaluate the systems’ usability (effectiveness, efficiency, and satisfaction), and security (confidentiality, integrity, and availability). The results are then used in Equation 5.10, where \( \alpha \) represents the overall usability evaluation, \( \beta \) represents overall security evaluation, \( \gamma \) represents the matrix score, which is the usable-security evaluation.

\[
\gamma = (\alpha \times \beta) - |\alpha - \beta| \tag{5.10}
\]
Figure 5.1: The Usable-Security Evaluation Matrix: The evaluation process consists of two components: formative mathematical-based modeling that evaluates the security and usability properties, and summative matrix that evaluates the usable-security based on the results of the formative modeling.

The highest measuring evaluation category in the matrix is when the system attains a score of at least 0.81 as displayed in Figure 5.2. This means that the system has applied and achieved high level of all of the six usability and security properties. Such a system is considered as usable and secure, not only because it achieved the usability and security goals, but because it bridged both usability and security with careful consideration to the HCI/user-centered approach from the beginning of the development process.

On the other hand, the lowest measuring evaluation category in the matrix is when a software system scores at most 0.01 as shown in Figure 5.2, which means that the system has not achieved any considerable values of the usability and security attributes. Such a system is considered neither usable nor secure, not only because it does not achieve the usability and security goals, but because it does not bridge usability and security as well. In addition, such a system indicates that its developers might have not had considered the HCI/user-centered approach.
Walking through the usable-security evaluation process, a final usable-security evaluation and categorization is reached as one of five levels: high usable-security is achieved when $\gamma \geq 0.81$; mostly usable-security is achieved when $0.49 \leq \gamma < 0.81$; usable-security is achieved when $0.36 \leq \gamma < 0.49$; some usable-security is achieved when $0.09 \leq \gamma < 0.36$; and usable-security is not achieved when $\gamma < 0.09$. Overall, a numerical categorization is given to each level as follows: high usable-security is categorized as 9, mostly usable-security is categorized as 7, usable-security is categorized as 6, some usable-security is categorized as 3, and not usable-security is categorized as 1. Figure 5.3 presents the final usable-security evaluation and categorization guide.
5.4 **Case Study: Authentication Approaches**

In order to show the advantage of using the usable-security evaluation matrix, this section presents the results of assessing the usability levels of an experiment that compares two authentication approaches: the Choice-Based Authentication Approach (CBAA) and the Traditional-Based Authentication Approach (TBAA) 60 scenarios (See Section 6.4). Sixty usable-security scenarios are used: thirty usability evaluation values representing the CBAA are considered for one set of 30 scenarios, while the other 30 usability evaluation values representing the TBAA are considered for another set of 30 scenarios.

For the security values, we used one value (mostly secure: 0.7) for all the 30 scenarios of the TBAA (the expected entropy is between $2^{26}$ and $2^{27}$). As the assumed security value is considered as the security level that can be achieved when the NIST SP 800 Series is used as a foundation security policy [103]. However, the security level of the CBAA has been increased because of the increased complexity for adversaries by displaying multiple authentication methods during the login process as explained in
Section 6.3 (the expected entropy is between $2^{64}$ and $2^{65}$, in addition to biometrics). Hence, we anticipated that the CBAA security should at least be better than the TBAA by 0.1. Therefore, we used (mostly secure: 0.8) as a unified security value for all the thirty scenarios of the CBAA. Tables 5.1, 5.2, and Figures 5.4 and 5.5 display the two sets of the scenarios.

According to the Final Usable-Security Evaluation and Categorization Guidance (See Figure 5.3), Table 5.1 of the CBAA shows that scenarios 21 and 25 provide some usable-security level (0.31 and 0.32, respectively), scenarios 3 and 6 provide usable-security level (0.40 and 0.48, respectively), while the rest of the 26 scenarios provide mostly usable-security level (i.e: the values are greater than or equals to 0.49, and less than 0.81). As a result, the overall usable-security evaluation value for the CBAA is 0.58, which indicates the mostly usable-security level.

The TBAA scenarios displayed on Table 5.2 show that scenarios 16 and 22 provide not usable-security level (0.09 and -0.19, respectively). Ten scenarios provide some
Table 5.1: The Scenarios of the CBAA. The First Set of Scenarios: 30 Scenarios on the Choice-Based Authentication Approach (CBAA)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Usability</th>
<th>Security</th>
<th>Usable-Security Per Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.84</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>2</td>
<td>0.82</td>
<td>0.80</td>
<td>0.64</td>
</tr>
<tr>
<td>3</td>
<td>0.66</td>
<td>0.80</td>
<td>0.40</td>
</tr>
<tr>
<td>4</td>
<td>0.79</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>5</td>
<td>0.74</td>
<td>0.80</td>
<td>0.53</td>
</tr>
<tr>
<td>6</td>
<td>0.71</td>
<td>0.80</td>
<td>0.48</td>
</tr>
<tr>
<td>7</td>
<td>0.89</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>8</td>
<td>0.90</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>9</td>
<td>0.75</td>
<td>0.80</td>
<td>0.55</td>
</tr>
<tr>
<td>10</td>
<td>0.93</td>
<td>0.80</td>
<td>0.61</td>
</tr>
<tr>
<td>11</td>
<td>0.74</td>
<td>0.80</td>
<td>0.54</td>
</tr>
<tr>
<td>12</td>
<td>0.89</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>13</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>14</td>
<td>0.85</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>15</td>
<td>0.89</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>16</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>17</td>
<td>0.82</td>
<td>0.80</td>
<td>0.64</td>
</tr>
<tr>
<td>18</td>
<td>0.90</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>19</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>20</td>
<td>0.86</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>21</td>
<td>0.62</td>
<td>0.80</td>
<td>0.31</td>
</tr>
<tr>
<td>22</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>23</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>24</td>
<td>0.86</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>25</td>
<td>0.62</td>
<td>0.80</td>
<td>0.32</td>
</tr>
<tr>
<td>26</td>
<td>0.80</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>27</td>
<td>0.83</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>28</td>
<td>0.90</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>29</td>
<td>0.92</td>
<td>0.80</td>
<td>0.62</td>
</tr>
<tr>
<td>30</td>
<td>0.88</td>
<td>0.80</td>
<td>0.62</td>
</tr>
</tbody>
</table>

Overall Usable-Security Evaluation | 0.58
Table 5.2: The Scenarios of the TBAA. The Second Set of Scenarios: 30 Scenarios on the Traditional-Based Authentication Approach (TBAA)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Usability</th>
<th>Security</th>
<th>Usable-Security Per Trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.55</td>
<td>0.70</td>
<td>0.24</td>
</tr>
<tr>
<td>2</td>
<td>0.77</td>
<td>0.70</td>
<td>0.46</td>
</tr>
<tr>
<td>3</td>
<td>0.77</td>
<td>0.70</td>
<td>0.46</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>0.70</td>
<td>0.35</td>
</tr>
<tr>
<td>5</td>
<td>0.88</td>
<td>0.70</td>
<td>0.43</td>
</tr>
<tr>
<td>6</td>
<td>0.90</td>
<td>0.70</td>
<td>0.42</td>
</tr>
<tr>
<td>7</td>
<td>0.86</td>
<td>0.70</td>
<td>0.43</td>
</tr>
<tr>
<td>8</td>
<td>0.82</td>
<td>0.70</td>
<td>0.45</td>
</tr>
<tr>
<td>9</td>
<td>0.76</td>
<td>0.70</td>
<td>0.46</td>
</tr>
<tr>
<td>10</td>
<td>0.77</td>
<td>0.70</td>
<td>0.46</td>
</tr>
<tr>
<td>11</td>
<td>0.87</td>
<td>0.70</td>
<td>0.43</td>
</tr>
<tr>
<td>12</td>
<td>0.61</td>
<td>0.70</td>
<td>0.33</td>
</tr>
<tr>
<td>13</td>
<td>0.75</td>
<td>0.70</td>
<td>0.47</td>
</tr>
<tr>
<td>14</td>
<td>0.55</td>
<td>0.70</td>
<td>0.23</td>
</tr>
<tr>
<td>15</td>
<td>0.80</td>
<td>0.70</td>
<td>0.45</td>
</tr>
<tr>
<td>16</td>
<td>0.46</td>
<td>0.70</td>
<td>0.09</td>
</tr>
<tr>
<td>17</td>
<td>0.51</td>
<td>0.70</td>
<td>0.16</td>
</tr>
<tr>
<td>18</td>
<td>0.59</td>
<td>0.70</td>
<td>0.30</td>
</tr>
<tr>
<td>19</td>
<td>0.73</td>
<td>0.70</td>
<td>0.47</td>
</tr>
<tr>
<td>20</td>
<td>0.65</td>
<td>0.70</td>
<td>0.41</td>
</tr>
<tr>
<td>21</td>
<td>0.72</td>
<td>0.70</td>
<td>0.48</td>
</tr>
<tr>
<td>22</td>
<td>0.30</td>
<td>0.70</td>
<td>-0.19</td>
</tr>
<tr>
<td>23</td>
<td>0.58</td>
<td>0.70</td>
<td>0.29</td>
</tr>
<tr>
<td>24</td>
<td>0.53</td>
<td>0.70</td>
<td>0.20</td>
</tr>
<tr>
<td>25</td>
<td>0.88</td>
<td>0.70</td>
<td>0.43</td>
</tr>
<tr>
<td>26</td>
<td>0.63</td>
<td>0.70</td>
<td>0.38</td>
</tr>
<tr>
<td>27</td>
<td>0.58</td>
<td>0.70</td>
<td>0.28</td>
</tr>
<tr>
<td>28</td>
<td>0.86</td>
<td>0.70</td>
<td>0.43</td>
</tr>
<tr>
<td>29</td>
<td>0.53</td>
<td>0.70</td>
<td>0.21</td>
</tr>
<tr>
<td>30</td>
<td>0.75</td>
<td>0.70</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Overall Usable-Security Evaluation 0.35
Figure 5.5: Radar Chart for the Usable-Security Evaluation Values. It displays usable-security coverage by the CBAA and the TBAA
usable-security level, and 18 scenarios provide usable-security level. Consequently, the overall usable-security evaluation value for the CBAA is 0.35, which indicates to the some usable-security level.

Figure 5.4 displays trend-lines of evaluating usable-security levels among all the scenarios of both the CBAA and the TBAA. The trend-line of the CBAA shows better achievement of usable-security than the trend-line of the TBAA, which indicates that the usable-security evaluation results are more stable in the CBAA than the TBAA.

Figure 5.5 displays a spider diagram (radar chart) that shows the usable-security coverage by both of the CBAA and the TBAA. As the diagram indicates that the CBAA occupies a wider usable-security domain than the TBAA.

5.5 Conclusion

Usability, security, and usable-security software quality attributes are important to be analyzed, specified within system’s requirements, properly designed, correctly constructed, accurately evaluated, and appropriately deployed. However, the previous methodology of applying the above attributes does not assure the development of perfect software systems with prefect quality standards. Applying the integration of such attributes becomes more important than just applying each attribute individually. This work goes beyond the application of attributes’ integration, as it proposes an evaluation methodology to test the alignment of those attributes. Moreover, the proposed evaluation matrix is flexible enough to modify the attributes’ measures, and also it is opened to add other quality attributes.
Chapter 6

Framework Evaluation: Heuristic and Comparative Experiment

Preface

Authentication is an important security component of almost any software application. It serves as the application’s first line of security by controlling access with the goal of protecting the confidentiality and integrity of the system. With the large variety of software applications that an end-user interacts with daily, authentication is becoming a usability issue that has the potential to weaken a system’s overall security. The increasing complexity of dealing with a variety of authentication mechanisms often causes end-users to develop negative security behaviors, such as writing down passwords. Some of the currently available authentication mechanisms, such as alphanumeric passwords, raise universal access issues due to both the issue of remembering a complex sequence of characters and the difficulty some individuals may have in entering that exact sequence on a keyboard or mobile device. This chapter
proposes an authentication approach that seeks to address usability, universal access, and security issues. The approach is called *Choice-Based Authentication Approach (CBAA)*.

In addition, we have conducted usable-security comparative experimental research that investigated the usable-security measures using the newly proposed CBAA approach. The approach is based on two concepts: allowing authentication according to end-user’s choice during the sign-up process for better usability and universal access, and increasing penetration complexity for adversaries by showing all the authentication methods during the login process for better security. The experiment compares between the proposed CBAA and the *Traditional-Based Authentication Approach - alphanumeric password (TBAA)*, in order to find out whether the new CBAA provides better usable-security than the TBAA.

### 6.1 Introduction

Computing systems have become some of the most important tools for easily exploring and investigating the intricate nature of art, sciences, and engineering. Humans interact with systems on a daily basis to get their jobs done efficiently and effectively. However, with the revolution of information technology, humans tend to use these systems with both good and bad intentions. Consequently, security and usability have become two important quality attributes that need to be adjusted, integrated, and properly balanced. Usability assumes that humans interact with systems in order to perform an appropriate task, while security considers that humans may interact with systems with malevolent intent. These two contradicting goals lead to a conflict of interest between the two attributes that is evident in irrational interaction design [11].
Human-Computer Interaction (HCI) specialists and security experts addressed this problem by forming a new solution field called usable-security [12]. Whitten and Tygar [105] define usable-security as a security system where users are aware of security tasks that need to be performed, are able to figure out how those tasks are properly performed, do not make harmful errors, and are comfortable with the interface. Usable-security has become a new hybrid software quality attribute with the goal of making security and usability synergistic rather than dissonant [43, 54, 91, 100]. To this end, HCI and security experts have provided various approaches and design techniques to merge, align, and integrate usable-security, such as user-centered interface design [15], early involvement of both security and usability [38], designation design [107], user decision-based security information [99], filtering users and transmitted data [72], using biometrics [71], using principles and patterns [43], and incorporating post-hoc security layers [46].

Authentication is one area that illustrates the security-usability conflict of interest [79]. Kumar [71] concluded that alphanumeric-password-based authentication cannot be both usable and secure at the same time and suggested graphical passwords and biometrics as alternatives to alphanumeric passwords. In contrast, Sasse et al. [92] concluded that alphanumeric passwords are not usable because usability is not considered as a fundamental security requirement.

Looking for solutions to the above problem, we have proposed a Choice-Based Authentication Approach (CBAA) with consideration of two concepts.

One concept comes from usability and universal access viewpoints which is based on giving systems’ user many options to choose from during the sign-up process. When the user is given the freedom to choose the most suitable method of interaction among various options to interact with the computer systems, he or she can be more satisfied
and feel more confident. The user’s cognitive load can also be decreased [17, 74, 64]. As a result, the user could implicitly be willing to cooperate with computer systems.

The second concept comes from a security viewpoint which is based on providing multiple interaction methods [102], so it becomes difficult for adversaries to know which method is chosen by the user. For example, user authentication systems usually provide one way for verifying users (user names and passwords, ID cards, tokens, or biometrics). In more advanced authentication systems, dual authentication is used to verify users (user names and passwords, and tokens or biometrics), however, the user still doesn’t have the choice to select the most suitable method of interaction with the system. It is true that dual authentication makes the complexity of security systems high enough to make it difficult to penetrate [102], but the question that comes is: is such authentication complexity necessary to make authentication systems difficult to penetrate? And is there a simpler way to make security systems difficult to penetrate rather than adding high complexity? [46].

It is worth noting that Microsoft Corporation adopted this concept on their current operating system "Microsoft Windows 8". Microsoft Windows 8 provides multiple authentication options such as the traditional alphanumeric password, picture password, personal identification number (PIN), and even fingerprint biometrics. However, despite the fact that the new ways of authentication for Microsoft Windows 8 enhanced the authentication usability, the security of such authentication can be enhanced through adopting the second concept and display all the authentication methods available on the login interface, just to make the login interface penetration even more complicated for adversaries.

To answer the above concerns, we have proposed the CBAA as a way to make security and usability synergetic in order to achieve an acceptable level of usable-
security [53]. The CBAA is developed based on the usable-security framework that we have been developing [51, 52]. The proposed approach needs to be tested through end-users.

This chapter evaluates the USEF and investigates whether the proposed approach of the CBAA - that demonstrates the USEF - raised security and provided usability and universal access, without adding more authentication complexity. Section 6.2 introduces heuristic evaluation for the USEF; Section 6.3 provides the CBAA as a case study demonstrating the application of the USEF; Section 6.4 provides an experiment that compares between the CBAA and the TBAA.

6.2 Heuristic Evaluation on The Framework

6.2.1 Overview

With a newly developed framework, we needed to get feedback on how robust and responsive it could be when tested in real world environments that consider assessing, integrating, and balancing security and usability as contradicting software quality attributes. Security practitioners, usability experts, software engineers, and information systems’ decision makers were all involved in this heuristic evaluation. As each of these disciplines play a partial role in the proposed framework. However, the primarily targeted group for evaluating the framework are those who specialized in both security and usability together, namely: usable-security experienced practitioners. Despite the unfortunate fact that usable-security professionals are very limited and hard to find, we were able to get the benefit of attending one of their most important events, HCI International Conference - HCII2014, at which usable-security experts gather together.
We were able to conduct a heuristic evaluation on the first two components of the framework at this conference.

The SDLC has five different phases, each phase has its own sub-group of developers, and all sub-groups work together in order to make one software development group. As stated in Chapter 1 that the framework triggers three phases of the SDLC, hence, requirements engineers, design engineers, software architects, and software testing and evaluation specialists were all involved in this survey as well.

### 6.2.2 Objectives

The survey on the framework is conducted in order to evaluate each of the first two framework’s components individually. The survey evaluated the integration between the two components when they work together as one usable-security processing framework. There were two primary goals to evaluate the success of the USEF framework:

1. How responsive the usable-security requirements’ assessment approach is to changes in requirements, and how accurately it measures the balance among the three attributes: security, usability, and usable-security during the requirements engineering phase of the SDLC.

2. How well the proposed usable-security design guidelines approach enhances the development of software products in terms of both linking the requirements to the design tools during the design phase of the SDLC, and integrating security and usability synergistically in order to have better usable-security.
6.2.3 Heuristic Evaluation Design

Our survey is designed using a two dimensional survey design (see Figure 6.1). The first is the quality attributes dimension, which consists of three parts: security, usability, and usable-security. The second is the SDLC phase dimension, which also consists of two parts representing the triggered phases of the SDLC: requirements and design. We should mention that the second dimension should include a third part, the evaluation/testing phase. However, the usable-security evaluation matrix, which is the third component of the USEF, was not ready during the conduction of the heuristic evaluation; and including it within the scope of our plan may delay our progress towards completing our degree project. Therefore, it will be considered for future work after completing the degree project.
Heuristics

The evaluation has 12 heuristics used to measure the first two components of the framework. The adopted heuristics are usually used to evaluate the newly proposed frameworks in order to validate their usability and ability to serve as approved scientific solutions that are developed based on a strong background, that are both usable and testable by intended users. The list of the heuristics along with their definitions is displayed as follows:

- $H_1$: **Adheres to good principles:** The process is based on fundamental and key principles.
- $H_2$: **Learnability:** Developers can easily learn the appropriate methods to operate the process.
- $H_3$: **Usability:** Developers can easily operate the process.
- $H_4$: **Documentation:** Developers can easily find, read, and understand the process documentations.
- $H_5$: **Maintainability:** Developers can modify the process to correct faults or adapt to environmental changes.
- $H_6$: **Upgradability:** Developers can raise the process to higher level of quality.
- $H_7$: **Scalability:** Developers can rely on the process to adapt with increased number of properties.
- $H_8$: **Robustness:** The process is resistive and able to cope with errors during its operation.
- $H_9$: **Flexibility**: The process is well responsive to changes.

- $H_{10}$: **Standards Usage**: The process adopts de-facto methods and policies.

- $H_{11}$: **Testability**: Developers can examine and evaluate the process in order to validate it.

- $H_{12}$: **Consistency**: The process provides harmonious agreement among its steps and its related processes.

**Evaluation Methodology**

The methodology of evaluating the heuristics adopts using a five level Likert scale [60]. Each of the first two components of the framework was evaluated by applying each of the 12 heuristics through ranking the heuristics based on the degree of agreement as shown in Table 6.1.

Table 6.1: The Ranking Terms and their values that each heuristic is ranked on in order to evaluate the framework components.

<table>
<thead>
<tr>
<th>Ranking Term</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>1</td>
<td>The heuristic under evaluation has not been met at all.</td>
</tr>
<tr>
<td>Disagree</td>
<td>2</td>
<td>The heuristic under evaluation has not been met, but might be met with some improvement.</td>
</tr>
<tr>
<td>Neutral</td>
<td>3</td>
<td>The heuristic under evaluation could not be evaluated, or the heuristic is not applicable.</td>
</tr>
<tr>
<td>Agree</td>
<td>4</td>
<td>The heuristic under evaluation has been met, but needs some enhancement.</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>5</td>
<td>The heuristic under evaluation has been satisfactorily met.</td>
</tr>
</tbody>
</table>
6.2.4 Heuristic Evaluation Results and Discussion

Results of the heuristic evaluation are displayed in Table 6.2. The overall result of the heuristic evaluation on the two first components of the usable-security engineering framework is $4.00 \approx 80\%$. The processes of the framework during the design phase, namely: the usable-security guidelines, has gotten higher evaluation value (4.60) than the work during the requirements engineering phase. This can be related to the fact that the work of the first component during the requirements engineering phase is based on techniques and tools brought from another field of science, decision science. The work of the second component during the design phase is straightforward and does not adopt ideas from other fields of science.

Table 6.2: The Final Results of the Heuristic Evaluation on the Usable-Security Engineering Evaluation Framework for Enhancing Software Development. D1 represents Quality Attributes’ Dimension, and D2 represents SDLC Phases’ Dimension

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D2 Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Security</td>
<td>Usability</td>
<td>Usable-Security</td>
</tr>
<tr>
<td>D2</td>
<td>Requirements</td>
<td>4.39</td>
<td>4.37</td>
</tr>
<tr>
<td></td>
<td>Design</td>
<td>4.68</td>
<td>4.67</td>
</tr>
<tr>
<td>D1 Average</td>
<td>4.53</td>
<td>4.52</td>
<td>4.33</td>
</tr>
</tbody>
</table>

The evaluation of the first component, the AFUS during the requirements engineering phase, comes out as follows:

1. The involvement of security requirements during the assessment process of the AFUS has been evaluated overall as 4.39 out of 5.00. This result indicates that the security requirements have been adequately considered and properly involved during the requirements assessment process.

2. The involvement of usability requirements during the assessment process of the AFUS has been evaluated overall as 4.37 out of 5.00. Similarly to the security
requirements, this evaluation result also indicates that usability requirements have been adequately considered and properly involved during the requirements assessment process. However, the involvement of the security requirements got a bit higher evaluation results (4.39) than the usability requirements (4.37).

3. The process of gathering initial usable-security requirements, and merging security and usability requirements together to generate new usable-security requirements, in order to integrate and align security and usability requirements together during the assessment process of the AFUS has been evaluated overall as 4.22 out of 5.00. Again, this evaluation value is still within the accepted range, as it resides in the area between the ”agree” and ”strongly agree” rankings, regardless of the fact that the process of involving usable-security requirements got the lowest evaluation result (4.22) comparing to the security and usability requirements: 4.39 and 4.37, respectively. These differences are logically and scientifically justifiable, as the research work on security and usability requirements is more robust and stable comparing to the work on the usable-security requirements.

The evaluation of the second component, which is the usable-security guidelines during the design phase, comes out as follows:

1. The process of associating security requirements with their appropriate design tools (principles, techniques, patterns, and best practices) through applying the usable-security guidelines during the design phase has been overall evaluated as 4.68 out of 5.00. This evaluation result indicates that the usable-security guidelines have adequately considered security and properly associate security requirements with the appropriate design tools.
2. The process of associating usability requirements with their appropriate design tools (principles, techniques, patterns, and best practices) through applying the usable-security guidelines during the design phase has been overall evaluated as 4.67 out of 5.00. This evaluation result indicates that the usable-security guidelines have adequately considered usability and properly associate usability requirements with the appropriate design tools.

3. The processes of associating usable-security requirements with appropriate design tools, and analyzing and resolving the trade-offs between security and usability design tools through applying the linkage mapping table, trade-off analysis, and the checkpoints during the design phase have been evaluated overall as 4.33 out of 5.00. This result is accepted in spite of its being the lowest result. Dealing with conflict objectives is not an easy task at all, however, this evaluation showed an improvement towards the goal of integrating security and usability and resolving the trade-offs between them.

The average abstract evaluation values of involving and considering security, usability, and usable-security on our usable-security engineering framework are 4.53, 4.52, and 4.33, respectively.

6.3 A Case Study: Choice-Based Authentication Approach (CBAA)

Typical authentication systems provide one method for verifying users identities (passwords, ID cards, tokens, or biometrics) [61]. In some advanced authentication systems two-factor authentication is used to verify users. Two-factor authentication is
a methodology that uses two different factors, such as combining passwords, tokens or biometrics together to authenticate individuals [61]. However, the users still do not have the option to select their most suitable method of interaction with the system. It appears that two-factor authentication increases the complexity of security systems enough to make penetration more difficult [12]. This observation prompts two questions: **is increased computational complexity necessary to make authentication systems difficult to penetrate?** And **is there a simpler way to make security systems simultaneously usable and difficult to penetrate rather than increasing the complexity?** [46]

As stated in Section 6.1, to address the above concerns, we propose a Choice-Based Authentication Approach (CBAA) as a way to make security and usability synergistic to achieve an acceptable level of usable-security without increasing users’ cognitive load.

As introduced above, the approach is based on two concepts. The first concept is allowing end-users to select their authentication method for better usability in order to decrease users’ cognitive load and increase their desire to cooperate with the system. In addition, this concept supports increased universal access because it allows users to select from a group of authentication approaches that are more suitable for a broad range of users [74].

The second concept is increasing the difficulty for adversaries by displaying all of the possible authentication methods during the login process. The goal of this work is not to produce a novel authentication method, but rather to demonstrate that it is possible to employ currently available authentication technologies in a way that is both more secure and more usable.
CBAA Demonstration

Figures 6.2 and 6.3 show an authentication system that is based on the CBAA approach. The system uses three different authentication methods: alphanumeric password based on the NIST SP 800 Series [18], graphical password based on the recall-based method [31, 90], and fingerprint biometrics [2, 61, 74, 100]. The purpose of the CBAA demonstration is to investigate whether usable-security can be achieved by giving end-users the freedom to select the authentication method they prefer. Each of the three authentication methods will be described in details as follows.

Alphanumeric Password.

For the purposes of this demonstration, we derived a simplified password creation policy that was adapted from the password strength guidelines provided by the National Institute of Standards and Technology (NIST SP 800 Series) [18]. For a real system both
The guidelines used for this approach consist of the following policy points:

1. The length of the password must be at least eight characters.
2. Must contain at least one digit (number).
3. Must contain at least one special character.
4. Must contain at least one capital letter.
5. Must contain at least one lower letter.

**Graphical Password.**

There are two main approaches for graphical passwords: recognition based and recall based [31, 90]. The recognition based approach relies on already created and displayed grids or objects that users select as passwords. The recall based approach...
has the user select specific points from a picture that is provided. The experiment in this paper employs the second approach, the recall based, because it is more secure against many attacks such as shoulder surfing and social engineering [90]. Recall based graphical passwords are also more difficult to guess than the recognition based approach [31].

This approach is concerned with user choices, so the security policy it uses for graphical passwords is less secure than should be used in real systems. Subjects who choose to use graphical authentication are provided with a picture and are asked to select point(s) from it based on the following guidelines:

1. The number of point(s) chosen must be between one and four.
2. Point(s) can be shosen from anywhere in the given picture.
3. Point(s) must be recalled in the same sequence as it was chosen.

**Biometrics.**

There are many biometric traits available for authentication, such as face, fingerprint, voice, iris, signature, gait, hand geometry, palm-print, and soft biometrics [61, 74]. Fingerprints are the most popular biometric trait that can easily be extracted in a controlled environment such as for lab experiments [2, 100]. The subjects who choose biometrics as an authentication method are provided with a commercially-available fingerprint sensor to scan their fingerprints.

**Heuristic Analysis and Evaluation**

Although the ultimate goal of proposing this approach is to demonstrate the USEF and conduct controlled experiments with a wide range of users, we first chose to
investigate the effectiveness of the proposed approach by conducting an informal usable-security heuristic evaluation. This heuristic evaluation was a critique-based expertise and heuristic feedback investigation that is performed by security and usability experts. There were four main advantages of conducting this type of evaluation [77]: 1) identifying minor issues to resolve them before involving end-users, so formal experiments focus on major design issues; 2) enhancing the design through obtaining improvement ideas that may be presented by the evaluators before conducting the official experiment; 3) providing indicative data that helps in identifying the possible directions and potential success of running formal experiments; and 4) refining the rough edges.

The informal usable-security heuristic evaluation for the demonstration of the proposed CBAA depended on investigating its compliance with the heuristic usability principles of Jakob Nielsen [76]. Moreover, the usable-security principles of Simson Garfinkle [43] were added to the list of the principles. The informal evaluation was done with several individuals. Seven experts, five males and two females, worked through the demonstration twice and went over all of the tasks during each run. The experts’ security and usability experience is between one and over 20 years. The following list represents the heuristics used for evaluating the demonstration:

- $H_1$ : **Visibility of System Status**: The system provides feedback on the status of performed.
- $H_2$ : **Match between System & World**: The system speaks the users’ language (meaningful vocabulary, phrases, and concepts).
- $H_3$ : **User Control & Freedom**: The system provides the choice of “exit” at any time.
• **H₃**: **Design Consistency & Security Standards**: The system provides enough consistency on flow control and objects placement, and follows standard policies.

• **H₅**: **Error Prevention**: The system itself encourages error avoidance by controlling the process flow.

• **H₆**: **Recognition Support**: The system supports recognition though minimizing recall (minimizing cognitive load).

• **H₇**: **Flexibility & Efficiency of Use**: The system allows skipping unnecessary or known steps (shortcuts) to save time.

• **H₈**: **Aesthetic & Minimalist Design**: The system provides only relevant information for each dialog.

• **H₉**: **Help Users Recognize, Diagnose, and Recover from Errors**: The system provides error messages that indicate the problems and suggest proper solutions.

• **H₁₀**: **Help & Documentation**: The system provides instant and informative help.

• **H₁₁**: **Least Surprise & Astonishment**: The system avoids surprising and astonishing users with unexpected information or actions.

• **H₁₂**: **Good Security Now**: The system adopts up-to-date security techniques.

• **H₁₃**: **No External Burden**: The system avoids affecting other systems and applications negatively.

**Results and Discussion**

Results of the heuristic evaluation are displayed in Figure 6.4. The heuristic evaluation levels ranged between 6.7 and 9.3 with an overall average evaluation level of 8.1.
Figure 6.4: The Results of Heuristic Evaluation
Both $H_2$ and $H_{12}$ received the highest evaluation level (9.3). This indicates that the CBAA system uses meaningful vocabulary, phrases, and concepts that speak end-users’ language. Moreover, the system adapts up-to-date security techniques by using graphical passwords and biometrics (fingerprints).

$H_1$, $H_3$, $H_4$, $H_5$, and $H_{11}$ were evaluated at levels between 8.4 and 8.1, which indicates that the system provided feedback on the status of user’s progress. Moreover, it allowed users to exit at any time, provided consistent flow control and object placement of the design along with following the standard authentication policies. It also encouraged error avoidance by controlling the process flow and avoided unexpected information and/or actions.

The rest of the heuristic points: $H_6$, $H_7$, $H_8$, $H_9$, $H_{10}$, and $H_{13}$, were evaluated at levels below 8, which indicated that more work is needed to enhance the usable-security of the proposed system by improving these heuristics. We concluded that the system should be enhanced in the following ways:

- Help end-users to recognize the authentication processes without having to recall much information or need experience to interact with the proposed authentication approach.

- Allow end-users to skip unnecessary steps through providing shortcuts, so users can move to the other authentication methods at any time.

- Focus on only providing relevant information for each dialog.

- Support end-users with helpful error messages that clearly explain the error and suggest solutions.

- Provide adequate help and documentation.
After analyzing the results of the heuristic evaluation, we revised the demonstration version of the CBAA to address the usability and security points that were shown to need further enhancement. The updated version of the interface was used for usable-security research that compares the usable-security level between the proposed CBAA and the currently available traditional-based authentication systems, to determine whether the CBAA provides better usable-security than the authentication systems that are currently available. A full-scale survey of sixty users has been conducted and the results were presented in Section 6.4.

6.3.1 Applying the AFUS during the Requirements Phase

As a detailed illustration for the first component of the usable-security framework, this part presents the application of the AFUS on the demonstration (CBAA). As Chapter 3 describes the AFUS in details, this part focuses on the application processes only based on Figure 3.1. Appendix B provides detailed explanation about applying the AFUS on the CBAA during the requirements engineering phase.

Requirements Filtering and Merging for the CBAA

In this phase, the stakeholders should meet together to derive the project requirements and analyze the generated requirements [98]. The scope of the AFUS is limited to the requirements that are related to security, usability, and usable-security. The system requirements are filtered to select and gather the requirements that are within the scope of the AFUS and then they are grouped into three main groups: security requirements, usability requirements, and usable-security requirements. The requirements of each of the three groups are filtered, merged, and rated according to their importance based
on the illustration of Section 3.2.1. The results of applying Section 3.2.1 on the three quality attributes (security, usability, and usable-security) are displayed on Tables 6.3, 6.4, 6.5, and 6.6, respectively.

This step is used to filter, rate, and merge the security and usability requirements; and initiate a new group of usable-security requirements. As mentioned earlier in Chapter 3, the typically available assessment methodology can focuses on one quality attribute at a time. For instance, OWASP Risk Rating Methodology [78] can only evaluate security levels for software systems, while SALUTA Attribute Preference Table [40] can only evaluate usability levels for security systems. As a result of evaluating both of the quality attributes separately using an adapted rating guidance form the OWSP RRM and the SALUTA APT (Section 3.2.1), security and usability requirements were assessed as it appears in Table 6.7.

Utility Functions for the CBAA

In order to evaluate usable-security, both security and usability requirements must be integrated together because usable-security is the intersecting point that the two quality attributes meet at. Unfortunately, security and usability conflict with each other at several points. In Chapter 3, we used the utility functions from the decision science field to utilize their ability in dealing with conflicting objectives such as the case of security and usability. In this section, we applied the utility functions (described in Section 3.2.2) to the framework demonstration (the CBAA) in order to dynamically assess security, usability, and usable-security requirements based on the process of the above section (Section 6.3.1).

First: Using the Individual Utility Function
Table 6.3: Security Requirements Filtering and Rating. This table displays the results of applying Section 3.2.1 on the security requirements group

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>CI</th>
<th>NI</th>
<th>AI</th>
<th>SecR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SReq₁</td>
<td>The system shall be available for use 24 hours per day, 365 days per year.</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3.33</td>
</tr>
<tr>
<td>SReq₂</td>
<td>The system shall only allow authorized users to access authorized data and resources.</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>4.67</td>
</tr>
<tr>
<td>SReq₃</td>
<td>The system shall only allow authorized users to alter authorized data and only on a proper way.</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>4.67</td>
</tr>
<tr>
<td>SReq₄</td>
<td>The system shall hide the details of its construction from the user.</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>4.67</td>
</tr>
<tr>
<td>SReq₅</td>
<td>The system shall provide a separate login interface.</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3.33</td>
</tr>
<tr>
<td>SReq₆</td>
<td>The system shall show all authentication methods on the login interface.</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>3.67</td>
</tr>
<tr>
<td>SReq₇</td>
<td>The system shall meet the NIST 800-Series.</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>4.00</td>
</tr>
<tr>
<td>SReq₈</td>
<td>The system shall not store users’ private information.</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>4.33</td>
</tr>
<tr>
<td>SReq₉</td>
<td>The system shall make the blind guessing attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33</td>
</tr>
<tr>
<td>SReq₁₀</td>
<td>The system shall make the focused guessing attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33</td>
</tr>
<tr>
<td>SReq₁₁</td>
<td>The system shall make the observation attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33</td>
</tr>
<tr>
<td>SReq₁₂</td>
<td>The system shall provide a separate sign-up interface.</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.67</td>
</tr>
<tr>
<td>SReq₁₃</td>
<td>The system shall provide least granting authority.</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>4.00</td>
</tr>
</tbody>
</table>

**Rating Totals**

<table>
<thead>
<tr>
<th></th>
<th>SHc</th>
<th>SHn</th>
<th>SHa</th>
<th>SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66</td>
<td>48</td>
<td>31</td>
<td>48.33</td>
</tr>
</tbody>
</table>
Table 6.4: Usability Requirements Filtering and Rating. This table displays the results of applying Section 3.2.1 on the usability requirements group

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>EI</th>
<th>FI</th>
<th>SI</th>
<th>UsaR</th>
</tr>
</thead>
<tbody>
<tr>
<td>UReq1</td>
<td>The system shall be effective by making the users able to complete all the required tasks.</td>
<td>9</td>
<td>3</td>
<td>7</td>
<td>6.33</td>
</tr>
<tr>
<td>UReq2</td>
<td>The system shall help the users to avoid making mistakes by providing clear steps to be followed.</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>3.67</td>
</tr>
<tr>
<td>UReq3</td>
<td>The system shall make the users satisfied and want to use it by adopting user-friendly interfaces.</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>5.00</td>
</tr>
<tr>
<td>UReq4</td>
<td>The system shall allow users to complete the required tasks in an acceptable amount of time.</td>
<td>3</td>
<td>9</td>
<td>7</td>
<td>6.33</td>
</tr>
<tr>
<td>UReq5</td>
<td>The system shall be as universally accessed as possible by providing at least three methods of authentication.</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>5.00</td>
</tr>
<tr>
<td>UReq6</td>
<td>The system shall not increase the users’ cognitive load through providing consistent steps and interfaces.</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6.00</td>
</tr>
<tr>
<td>UReq7</td>
<td>The system shall provide an acceptable sign-up time that does not exceed the standard average sign-up time.</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>5.33</td>
</tr>
<tr>
<td>UReq8</td>
<td>The system shall provide an acceptable login time that does not exceed the standard average login time.</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>5.33</td>
</tr>
<tr>
<td>UReq9</td>
<td>The system shall be easy to use by making the users self-helpers and not need to guidance to use it.</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6.00</td>
</tr>
<tr>
<td>UReq10</td>
<td>The system shall not confuse the users by avoiding duplication, double vision, implicit negatives, dangling alternatives.</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>6.00</td>
</tr>
<tr>
<td>UReq11</td>
<td>The system shall provide complete freedom to the users to choose whichever authentication method they prefer.</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4.00</td>
</tr>
<tr>
<td>UReq12</td>
<td>The system shall provide comfortable ways for the users to perform tasks.</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>6.00</td>
</tr>
</tbody>
</table>

Rating Totals

<table>
<thead>
<tr>
<th>SHE</th>
<th>SHf</th>
<th>SHs</th>
<th>SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>58</td>
<td>75</td>
<td>65</td>
</tr>
</tbody>
</table>
Table 6.5: The Initial Usable-Security Requirements Filtering and Rating. This table displays the results of applying Section 3.2.1 on the initial usable-security requirements group.

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>I</th>
<th>$I_{Usa_SecR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$IReq_1$</td>
<td>The system shall consider the stakeholders from the beginning of development.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_2$</td>
<td>The system shall make security and usability synergetic.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_3$</td>
<td>The system shall consider the HCI principles.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_4$</td>
<td>The system shall consider the HCI-SEC principles.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_5$</td>
<td>The system shall make its users know how to properly perform the tasks by making the interfaces expressiveness.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_6$</td>
<td>The system shall make all of its features visible to the users.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_7$</td>
<td>The system shall make the consequences of the decisions that the users make clearly indicated.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_8$</td>
<td>The system shall make its users aware of its information practices before collecting data from them.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_9$</td>
<td>The system shall notify customers of changes to its information policy.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{10}$</td>
<td>The system shall help the user to avoid making mistakes by supporting identifiability.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{11}$</td>
<td>The system shall provide a path of least resistance.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{12}$</td>
<td>The system shall use symbols and words that are naturally understandable by the user community.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{13}$</td>
<td>The system shall provide inherent connection between a user interface and its functional and physical properties.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{14}$</td>
<td>The system shall allow the users to be able to accomplish a task with a minimum of interaction steps.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{15}$</td>
<td>The system shall provide the users with useful feedback in a timely and relevant way.</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{16}$</td>
<td>The system shall provide varieties of methods that provide both of recognition-based and recall-based. (Memory)</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{17}$</td>
<td>The system shall provide mechanisms that are forgiving for recognition-based methods. (Memory)</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{18}$</td>
<td>The system shall provide mechanisms that are relatively strict for recall-based methods. (Memory)</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$IReq_{19}$</td>
<td>The system shall make both security and usability as operation tasks as possible rather than supporting tasks.</td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>

Rating Totals

<table>
<thead>
<tr>
<th>SH</th>
<th>$I_{Usa_SecR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>171</td>
<td>114</td>
</tr>
</tbody>
</table>
Table 6.6: The Merged Usable-Security Requirements Rating. This table displays the results of applying Section 3.2.1 on the merged usable-security requirements group

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>CSR</th>
<th>$\sum_{SecR}$</th>
<th>CUR</th>
<th>$\sum_{UsaR}$</th>
<th>$MU_{sa_SecR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{Req1}$</td>
<td>The system shall use symbols and words that are naturally understandable by the user community to prevent incorrect results.</td>
<td>2</td>
<td>8.67</td>
<td>2</td>
<td>9.67</td>
<td>3.06</td>
</tr>
</tbody>
</table>

**Rating Totals**

<table>
<thead>
<tr>
<th>CSR</th>
<th>$\sum_{SecR}$</th>
<th>CUR</th>
<th>$\sum_{UsaR}$</th>
<th>$MU_{sa_SecR}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8.67</td>
<td>2</td>
<td>9.67</td>
<td>3.06</td>
</tr>
</tbody>
</table>

Table 6.7: Assessing Security and Usability Requirements based on the OWASP RRM and the SALUTA APT. At this point, usable-security requirements have not been assessed

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Security</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment Value</td>
<td>0.426</td>
<td>0.574</td>
</tr>
</tbody>
</table>

- The ratios of the properties of security and usability attributes are also calculated as described in Section 3.2.2. The results are shown in Tables 6.8 and 6.9.
- The resulted ratios of the attributes’ properties are used for pointing out (The starting value for the pointing out was five). The results are shown in Tables 6.10 and 6.11.
- After calculating the ratios and the pointing outs for the properties of both security and usability, the weights of such properties are calculated based on

Table 6.8: The Ratios of the Security Properties (Confidentiality, Integrity, and Availability)

<table>
<thead>
<tr>
<th>R</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>2.13</td>
</tr>
<tr>
<td>Integrity</td>
<td>1.55</td>
</tr>
<tr>
<td>Availability</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Table 6.9: The Ratios of the Usability Properties (Effectiveness, Efficiency, and Satisfaction)

<table>
<thead>
<tr>
<th>R</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>1.07</td>
</tr>
<tr>
<td>Efficiency</td>
<td>1.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>1.29</td>
</tr>
</tbody>
</table>

Table 6.10: The Pointing out Process of the Security Properties (Confidentiality, Integrity, and Availability)

<table>
<thead>
<tr>
<th>P</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>10.65</td>
</tr>
<tr>
<td>Integrity</td>
<td>7.74</td>
</tr>
<tr>
<td>Availability</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6.11: The Pointing out Process of the Usability Properties (Effectiveness, Efficiency, and Satisfaction)

<table>
<thead>
<tr>
<th>P</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>5.35</td>
</tr>
<tr>
<td>Efficiency</td>
<td>5</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>6.47</td>
</tr>
</tbody>
</table>
Table 6.12: The Weights of the Security Properties (Confidentiality, Integrity, and Availability)

<table>
<thead>
<tr>
<th>K</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>0.46</td>
</tr>
<tr>
<td>Integrity</td>
<td>0.33</td>
</tr>
<tr>
<td>Availability</td>
<td>0.21</td>
</tr>
</tbody>
</table>

Table 6.13: The Weights of the Usability Properties (Effectiveness, Efficiency, and Satisfaction)

<table>
<thead>
<tr>
<th>K</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>0.32</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.30</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.38</td>
</tr>
</tbody>
</table>

The resulted values of their ratios and pointing outs. The formulas used to derive the weights are displayed in Section 3.2.2. The resulted weights are presented in Tables 6.12 and 6.13.

- The utilities of the properties of both security and usability are calculated using the values of their ratios, pointing outs, and weights. The individual utility function is used to perform the calculation of the properties’ utilities as described in Section 3.2.2. Tables 6.14 and 6.15 display the resulted utility values for the properties.

Second: Using the Additive Utility Function

Table 6.14: The Utility Values of the Security Properties (Confidentiality, Integrity, and Availability)

<table>
<thead>
<tr>
<th>U</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>1.00</td>
</tr>
<tr>
<td>Integrity</td>
<td>0.49</td>
</tr>
<tr>
<td>Availability</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Table 6.15: The Utility Values of the Usability Properties (Effectiveness, Efficiency, and Satisfaction)

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>0.24</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 6.16: The Ratios of the Quality Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>1.00</td>
</tr>
<tr>
<td>Usability</td>
<td>1.34</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>2.42</td>
</tr>
</tbody>
</table>

- The ratios of the three quality attributes: security, usability, and usable-security are calculated as described in Section 3.2.2. The resulted ratios are displayed in Table 6.16.

- After calculating the quality attributes’ ratios, the weights of the attributes are calculated based on the resulted ratios. The formulas used to derive the weights are displayed in Section 3.2.2. The resulted weights are presented in Table 6.17.

- The utilities of the quality attributes are calculated using the values of their ratios and weights. The additive utility function is used to perform the calculation of the attributes’ utilities as described in Section 3.2.2. Table 6.18 displays the resulted utility values for the properties.

Table 6.17: The Weights of the Quality Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>0.21</td>
</tr>
<tr>
<td>Usability</td>
<td>0.28</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.51</td>
</tr>
</tbody>
</table>
Table 6.18: The Utilities of the Quality Attributes

<table>
<thead>
<tr>
<th>U</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>0.62</td>
</tr>
<tr>
<td>Usability</td>
<td>0.46</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Table 6.19: Assessment for the CBAA

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Assessment</th>
<th>Security</th>
<th>Usable-Security</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBAA</td>
<td>Before applying AFUS</td>
<td>0.426</td>
<td>No Assessment</td>
<td>0.574</td>
</tr>
<tr>
<td></td>
<td>After applying AFUS</td>
<td>0.246</td>
<td>0.508</td>
<td>0.247</td>
</tr>
</tbody>
</table>

Decision Tree for the CBAA

The utility functions calculated weights and utilities of each quality attribute along with calculating weights and utilities of each attribute’s properties. Consequently, all of the minute details of all attributes and their properties are available. These details can be used to assess the attributes interactively. The gap between security and usability attributes is considered when assessing usable-security, because achieving usable-security is negatively correlated with the distance value between security and usability assessments. The narrower the gap between security and usability assessments, the better usable-security value can be achieved. In Section 3.2.3 we discussed the security usability gap issue and described a way of using such a gap in assessing usable-security. All of the minute details of the attributes are displayed on the decision tree in Figure 6.5. The assessment results that are derived from the decision tree are shown in Table 6.19.
Figure 6.5: The Decision Tree for the CBAA: it displays all the details of the quality attributes (security, usability, and usable-security) along with the details of their properties
6.3.2 Applying the Guidelines during the Design Phase

Applying the usable-security guidelines on the demonstration of the framework is a continual process of a further stage of the usable-security framework. This step can be used as a component of the framework following the AFUS process. Although the guidelines are set to be flexible enough to be used individually and are portable enough for use with other assessment methodologies. In this part of the case study, the guidelines are used as a component of the proposed usable-security framework and they rely on the result of applying the AFUS on the CBAA as shown in the following sections.

Selection of priority sensitivity

As described earlier in Chapter 4, this step relies on the requirements engineering phase of the SDLC. In order to identify the target priority levels, we used the results of rating the requirements for CBAA based on the AFUS (see Tables 6.20 and 6.21). All rating values are to be selected for higher sensitivity, the rating values starting from 6 are to be selected for medium sensitivity, and the rating values starting from 7 are to be selected for the lower sensitivity level. We adopted the generalization strategy that applies one design goal over all the requirements and we used the medium level of priority sensitivity, which means that priority values starting from 6 should be associated with appropriate design tools. These parameters were used specifically to present a typical demonstration of the guidelines.
Table 6.20: Security Requirements Filtering

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SReq1</td>
<td>The system shall be available for use 24 hours per day, 365 days per year.</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SReq2</td>
<td>The system shall only allow authorized users to access authorized data and resources.</td>
<td>7</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>SReq3</td>
<td>The system shall only allow authorized users to alter authorized data and only on a proper way.</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>SReq4</td>
<td>The system shall hide the details of its construction from the user.</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>SReq5</td>
<td>The system shall provide a unique login interface.</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SReq6</td>
<td>The system shall show all authentication methods on the login interface.</td>
<td>7</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>SReq7</td>
<td>The system shall meet the NIST 800-Series.</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SReq8</td>
<td>The system shall not store users’ private information.</td>
<td>6</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>SReq9</td>
<td>The system shall make the blind guessing attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SReq10</td>
<td>The system shall make the focused guessing attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SReq11</td>
<td>The system shall make the observation attack difficult to occur.</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SReq12</td>
<td>The system shall provide a separate sign-up interface.</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>SReq13</td>
<td>The system shall provide least granting authority.</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
Requirements filtering

Based on the sensitivity targets from the previous step (i.e., “medium” sensitivity), the requirements are filtered to identify any that are not expected to be associated with a design tool. Table 6.20 shows a list of security requirements, where Confidentiality, Integrity, and Availability represent the security properties [80]. At least one of those three properties must meet the expected priority level to be assigned a design tool. For example, in Table 6.20, $SReq_{12}$ will be eliminated because it doesn’t have rating values of 6 or higher and is considered to be a low priority requirement. Similarly, Table 6.21 shows the of usability requirements, where Effectiveness, Efficiency, and Satisfaction represent the usability properties [59].

Selection of appropriate design tools

For this case, all of the available design tools are considered. Therefore, best practices, techniques, activities, and patterns can all be assigned to requirements during the design phase.

Creation of a linkage mapping table

After completion of the above steps of the guideline, a linkage mapping table can be created through associating each requirement with appropriate design tools. For example, the first security requirement $SReq_5$ on Table 4.5 mandates providing a unified individual login interface that all users should use during the login process. This requirement can be met through applying two well-known security patterns: SINGLE ACCESS POINT [108] and FRONT DOOR [39]. In the same way usability requirements are associated with appropriate design tools as shown in Table 6.23. We provided
Table 6.21: Usability Requirements Filtering

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UReq_1$</td>
<td>The system shall be effective by making the users able to complete all the required tasks.</td>
<td>9</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>$UReq_2$</td>
<td>The system shall help the users to avoid making mistakes by providing clear steps to be followed.</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$UReq_3$</td>
<td>The system shall make the users satisfied and want to use it by adopting user-friendly interfaces.</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>$UReq_4$</td>
<td>The system shall allow users to compete the required tasks in an acceptable amount of time.</td>
<td>3</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>$UReq_5$</td>
<td>The system shall be as universally accessed as possible by providing at least three methods of authentication.</td>
<td>6</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_6$</td>
<td>The system shall not increase the users’ cognitive load through providing consistent steps and interfaces.</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_7$</td>
<td>The system shall provide an acceptable sign-up time that does not exceed the standard average sign-up time.</td>
<td>1</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_8$</td>
<td>The system shall provide an acceptable login time that does not exceed the standard average login time.</td>
<td>1</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_9$</td>
<td>The system shall be easy to use by making the users self-helpers and not need to guidance to use it.</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_{10}$</td>
<td>The system shall not confuse the users by avoiding duplication, double vision, implicit negatives, dangling alternatives.</td>
<td>9</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_{11}$</td>
<td>The system shall provide complete freedom to the users to choose whichever authentication method they prefer.</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$UReq_{12}$</td>
<td>The system shall provide comfortable ways for the users to perform tasks.</td>
<td>6</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>
examples of the design tools used to meet the high priority requirements of Tables 6.20 and 6.21.

$DT_1$: SINGLE ACCESS POINT [108]

$DT_2$: FULL VIEW WITH ERROR [108].

$DT_3$: SECURITY POLICY ENFORCEMENT [48].

$DT_4$: ANONYMITY SET [48].

$DT_5$: FORM/FIELD VALIDATION [39].

$DT_6$: PROGRESS INDICATION [39].

$DT_7$: MULTI-TASKING [39].

$DT_8$: EMULATION [39].

$DT_9$: ALERTS [39].

$DT_{10}$: SHORTCUTS [39].

$DT_{11}$: COGNITIVE WALK-THROUGH [33].

$DT_{12}$: ACTIONS FOR MULTIPLE OBJECTS [39].

$DT_{13}$: HISTORY LOGGING [39].

$DT_{14}$: PROTOTYPING [33].

$DT_{15}$: WIZARD [39].

**Generation of new design tools**

This is an optional step and added to the guideline steps in order to make the guideline flexible enough for design tools development and generation. However, there are enough design tools that meet the requirements of our authentication system, therefore, there is no need to generate new tools in this case.
Table 6.22: The Linkage Mapping Table for the Security Requirements

<table>
<thead>
<tr>
<th>Security Requirements Rating</th>
<th>Requirements</th>
<th>CONF</th>
<th>INT</th>
<th>AVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>SReq5</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>SReq6</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>SReq7</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>SReq8</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.23: The Linkage Mapping Table for the Usability Requirements

<table>
<thead>
<tr>
<th>Usability Requirements Rating</th>
<th>Requirements</th>
<th>EFF1</th>
<th>EFF2</th>
<th>SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>UReq2</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq3</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq4</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq6</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq7</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq8</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq10</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq9</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rate</td>
<td>UReq10</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Design Tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.24: Design Tools Trade-Off Matrix for the CBAA Demonstration

<table>
<thead>
<tr>
<th></th>
<th>DT2</th>
<th>DT_{12}</th>
<th>DT3</th>
<th>DT_{11}</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT_{12}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DT_{11}</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Analysis of the trade-off matrix**

The trade-off matrix analysis is used to identify any conflicts between the design tools selected to address the requirements’ needs during the design phase. Table 6.24 shows examples of conflicting cases where the adopted design tools work against one another. For instance, the **FULL VIEW WITH ERROR** pattern [108] conflicts with the usability activity for avoiding dangling alternatives and double vision that is **ACTIONS FOR MULTIPLE OBJECTS** [39]. Another conflicting case appears when the **SECURITY POLICY ENFORCEMENT** pattern [48] is used to ensure the application of the security policy, that may conflict with the **Cognitive Walk-through** [33] usability technique. These conflicting design tools have been identified and analyzed to allow us to address the conflicts. In the case of our authentication the conflict between design tools is resolved by adopting usable-security design tools such as **Interaction Design activities** [33], and **LEAST SURPRISE / LEAST ASTONISHMENT** and **PROVIDE STANDARDIZED SECURITY POLICIES** [43].

**Checking the match between requirements and design tools**

After refining and completing the linkage mapping tables, a matching check is performed to make sure that the requirements have been completely covered by the appropriate design tools and that there are no missing tools. A missing tool indicates
that the design phase is limited and therefore unable to properly meet the needs of all of the high priority requirements. In our demonstration we could successfully cover all the requirements by assigning each requirement with the appropriate design tools. The conflicts have been resolved by applying usable-security principles and patterns [43].

**Incorporation of results and feedback**

After a successful analysis the completed linkage mapping tables are ready to be passed to the architecture engineers so they can use them to select the appropriate design tools during the architecture design.

**6.3.3 Applying the Measuring Matrix during the Evaluation Phase**

Applying the measuring matrix to evaluate the achieved usable-security on the CBAA was shown in Section 5.4 when the CBAA used as a case study to illustrate the evaluation process. As the measuring matrix evaluated the usable-security level of the CBAA as 0.58, which indicates that the CBAA achieves the targeted level of usable-security: very usable-security.

**6.4 A Usable-Security Comparative Study: CBAA and TBAA**

**6.4.1 Concept of Experiment Operation**

The experiment has one ultimate goal to be achieved, to determine whether the CBAA helps achieving usable-security when end-users are given the freedom to select the authentication method they prefer. To illustrate the whole process of investigating such
a goal, we adapted the diagram of the essential process of developing insight from raw
data shown in [62]. The process started by collecting raw data, such as task completion
time, task completion status, error rate, degree of happiness, and degree of confusion.
After collecting the raw data statistical analysis was used to convert the raw data into
valuable statistical information such as mean, median, variance, standard deviation,
frequency, and percentage. Finally, the statistical information was converted to insight
where the right decision could be trustfully and safely made on promoting the thoughts
to become facts. Figure 6.6 depicts the concept of the experimental operation. As
this experiment incorporates human-subjects, an approval from the Institutional Review
Board must be gained before start working on the experiment. The Institutional Review
Board form along with its approval are displayed in Appendix C.

6.4.2 Hypotheses

Null Hypothesis

- $H_0$: There is no difference in the usable-security component and individual
  measures between the choice-based security approach and the traditional-based
  security approach when used by end-users.

Alternative Hypotheses

- $H_1$: The choice-based security approach has a higher task completion rate than
  the traditional-based security approach when used by end-users.

- $H_2$: The choice-based security approach needs a longer signing up time than the
  traditional-based security approach when used by end-users.
Figure 6.6: Concept of Experiment Operation for the Comparative Study between Choice-Based and Traditional-Based Authentication on Usability
• $H_3$: The choice-based security approach needs a shorter signing in time than the traditional-based security approach when used by end-users.

• $H_4$: The choice-based security approach has a higher degree of positive affect than the traditional-based security approach when used by end-users.

• $H_5$: The choice-based security approach is easier to use than the traditional-based security approach when used by end-users.

• $H_6$: The choice-based security approach has a better effectiveness level than the traditional-based security approach when used by end-users.

• $H_7$: The choice-based security approach has a better efficiency level than the traditional-based security approach when used by end-users.

• $H_8$: The choice-based security approach has a better satisfaction level than the traditional-based security approach when used by end-users.

• $H_9$: There is a difference between the choice-based security approach and the traditional-based at the component and individual measures of usable-security when used by end-users.

### 6.4.3 Variables

**Independent Variables**

There is only one independent variable for this experiment: security approach. The security approach independent variable has two levels:

**First Level**

Choice-based approach.
Second Level

Traditional-based approach.

Dependent Variables

The dependent variables of the experiment are the usable-security properties that are initially the standard usability properties (as component dependent variables) and their measures (as individual dependent variables) on security mechanisms. The following lists both of the individual and component dependent variables.

1. Individual Dependent Variables

(a) Number of Errors ($Error$): It is a dependent variable that counts the number of wrong steps conducted to complete a task [101, 59] (clicked links to perform a task, written passwords for alphanumeric passwords, selected objects for graphical passwords, or finger positioning for fingerprint biometrics).

The number of errors ($Error$) is measured by the percentage of errors rate per session out of the largest errors rate. The default value of $Error$ is 1, because it is important to avoid dividing by zero (0). A value of 0.2 is added to the error rate when an error occurs. For example: when two errors occurred during a particular session, the error rate is 1.4 (1+0.2+0.2).

(b) Task Completion ($Completion$): is the completion status of performing a task by a subject [59, 67]. It is measured by percentage.

The value of $Completion$ is a percentage between one hundred (100 percent) when all of the task parts are successfully completed, and zero (0 percent) when no part of the task is successfully completed.
(c) Sign-up Time \((Signup\_Time)\): is the time that a subject spends to complete the signing-up task during authentication creation and sitting up, measured in seconds.

(d) Login Time \((Login\_Time)\): It is the time that a subject spends to complete the signing in task (during the authentication request process), measured in seconds.

(e) Degree of Positive Affect \((Happiness)\): this dependent variable investigates how happy a subject is using the system, measured in percentage.

(f) Ease of Use \((Easiness)\): measures a subject’s feedback on how easy the approach is. It is measured in percentage.

(g) Degree of Negative Affect \((Clarity)\): This dependent variable investigates the confusion about using the system. The higher the clarity the system has, the lower the confusion. \(Clarity\) is also measured in percentage.

2. **Component Dependent Variables**

(a) Effectiveness: is the subject’s ability to reach his or her goal of performing a particular task completely and without errors. It is evaluated through two of the above individual dependent variables, task completion \((Completion)\), and number of errors \((Error)\). The justification for using these two individual dependent variables for effectiveness comes from the fact that errors have a negative effect on completing a particular task properly, as task completion represents the usability, and the number of errors represents the affect. Therefore, the effectiveness is a result of dividing the task completion rate by the error rate. The following shows the effectiveness calculation:
\[ Effectiveness = \frac{Completion}{Error} \]  

(6.1)

(b) Efficiency: is the subject’s ability to perform a specific task and complete it within an acceptable amount of time evaluated by counting the time spent to complete the tasks. Two of the above dependent variables are used to calculate the efficiency. The first dependent variable is the signing up time, \textit{Signup Time}, the second independent variable is the logging in time, \textit{Login Time}. Using these two variables to calculate the efficiency is justified by the fact that both of the times are important to be considered as an accumulative task completion time. The efficiency is a result of adding up the two times, the signing up time and the logging in time [73]. The following shows the efficiency calculation:

\[ Efficiency = Signup\ Time + Login\ Time \]  

(6.2)

(c) Satisfaction: is the component dependent variable that investigates several aspects of a subject’s attitude, including affect, usability, and cognitive load. It is evaluated by providing a questionnaire to the end-users that combines all of the satisfaction aspects under investigation. The above individual dependent variables of satisfaction are: 1) the degree of the positive affect, \textit{Happiness}; 2) the ease of use, \textit{Easiness}; and 3) the degree of the negative affect, \textit{Confusion} [67, 101, 59]. The satisfaction is a result of multiplying the degree of affect with the ease of use, and the result is divided by the degree of negative effect. We use a well-known satisfaction survey called ”System Usability Scale SUS” (see Figure 6.7) that was developed by John Brooke
in 1986 [101], as the questionnaire covers the three individual dependent variables of satisfaction. The questionnaire gives a value out of a 100 to represent the satisfaction level.

6.4.4 Experiment Structure

Basic design is usually considered when there is only one independent variable [73]. The intersection effects between the levels of the independent variable and the dependent...
variables are studied in order to determine whether the dependent variables affect the usable-security level in both choice-based and traditional-based authentication approaches’ interfaces.

### 6.4.5 Design Type

For basic design structure, the experiment can be either between-group or within-group. Between-group design is the best design type that can be used [73], because it avoids the possible learning effect and does not take long time. Each subject is only exposed to one security approach. This offers the opportunity to study the usable-security of each approach. Figure 6.8 depicts the design type.
6.4.6 Experiment Description

There are two authentication interfaces, and each subject must be exposed to only one interface randomly. The first authentication interface represents the choice-based authentication approach, while the second represents the typical traditional-based authentication. The experiment is divided into two parts. The first part is a set of procedural and practical steps, and the second part is a user satisfaction survey. Next, we explain each part.

**Procedural Steps**

The first part is procedural and is performed in two different stages: initial stage and final stage. The following describe both of the stages:

1. The initial periodical stage is setting up the authentication’s profile, as the subjects are given two tasks to perform within a ten minutes time-frame:

   (a) Signing-up.

   (b) Logging-in.

2. During this final periodical stage, the subjects are asked to only perform one particular task within a five minutes time-frame:

   (a) Logging-in.

**User Satisfaction Survey**

The second part is a survey completed by the subjects to obtain their satisfaction level on both of the proposed choice-based and traditional-based approaches. The survey is a collection of Likert scale sections [60]. The duration of this part is five minutes. The result of this survey evaluates the degree of satisfaction of the subjects.
Participants’ Contribution and Time Duration

The total time of the experiment is estimated to be thirty (30) minutes per section. Table 4 explains the experimental duration. The two letters: P and S represent the contribution to a task. P means primary contributor, while S means secondary contributor.

### 6.4.7 Participants

The targeted subjects are undergraduate and graduate students from different disciplines. The reason for targeting students from different disciplines is to have enough subjects representing various security knowledge levels. Other types of the participants are the observers, the experimenters, and the development team [85]. The number of the subjects is 30 per each condition (the total number is 60 subjects).

### 6.4.8 Statistical Analysis

The experiment design is a between-group type. The analysis techniques used in this analysis are: the mean; the median; the standard deviation; and grouping and aggregation [62]. The collected data are normally distributed. In addition, the dependent
variables between the two approaches are equally scaled by intervals; therefore, parametric statistical analysis techniques are used, such as t-Tests, F-Tests (Analysis of Variance–ANOVA), and correlation analysis. As a between-group design, Two-Tailed Independent-Samples Significance Test (t-Test) and one-way analysis of variance (ANOVA) can be used as statistical analysis techniques. The Two-Tailed Independent-Samples Significance Test (t-Test) is best suited for this experiment rather than the others [73]. However, the one-way analysis of variance (ANOVA) is used to confirm the Two-Tailed Independent-Samples Significance Test (t-Test). Moreover, the correlation statistical analysis is used because it supports investigating the relationships between the variables in both approaches.

**Two-Tailed Independent-Samples Significance Test (t-Test)**

The Two-Tailed Independent-Samples Significance Test (t-Test) is a well-known statistical analysis method used to compare the means of exactly two groups to determine if there is significant difference between them [73, 60]. In our case we have two groups (conditions) that we need to compare between them: our proposed CBAA and the typically available TBAA. In order to compare the means of the variables between both the approaches in a proper manner, the Two-Tailed Independent-Samples Significance Test (t-Test) is used as the primary statistical significance test.

**One-Way Analysis of Variance-ANOVA (F-Test)**

ANOVA is also a well-known statistical analysis method used to compare the means of more than one group to figure out if there is significant difference between the groups [73, 60]. In our case, two groups (conditions) were compared: The CBAA
and the TBAA. This test is used as a secondary statistical significance test in order to confirm the t-Test results.

### 6.4.9 Results and Discussion

As explained in Section 6.4.8, we have conducted various parametric statistical analysis on the collected data. A Two-Tailed Significance Test (t-Test), a One-Way Randomized Analysis of Variance (F-Test). Both tests were conducted in order to investigate whether the null hypothesis is supported by the results of formulating both of the simple and complex dependent variables of usable-security on both approaches (CBAA and TBAA). As a result of conducting the above parametric statistical analysis, the following sections describe each of the two significance tests (t-Test and F-Test).

#### Two-Tailed Independent-Samples Significance Test (t-Test)

According to the results of t-scores on all of the dependent variables (simple and complex) as shown on Tables 6.26 and 6.27. Task Completion, Completion, is the only variable that does not statistically support rejecting the null hypothesis because most of the participants completed their sessions successfully. For the rest of the dependent variables, they are all supporting the rejection of the null hypothesis because all of their t-scores are beyond the critical value of 2.000 [60].

Therefore, the overall result of the t-Test on Usable-Security is: \( t(58) = 4.54, p < .05 \) (see Table 6.28). Based on the displayed result, the null hypothesis assuming that "there is no difference in the usable-security component and individual measures between the choice-based security approach and the traditional-based security approach when used by end-users" is rejected, and the alternative hypotheses are supported.
Table 6.26: Results of Conducting the Two-Tailed Independent-Samples Significance Test (t-Test) on Simple Variables: task completion (Completion); number of errors (Errors); sign-up time; login time; degree of positive affect (Happiness); ease of use(Ease); and degree of negative affect (Clarity)

<table>
<thead>
<tr>
<th>df=58</th>
<th>Simple Variables</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
<td>Satisfaction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Completion</td>
<td>Errors</td>
<td>Sign-up</td>
<td>Login</td>
<td>Happiness</td>
<td>Ease</td>
</tr>
<tr>
<td>t</td>
<td>0.59</td>
<td>-2.94</td>
<td>-2.61</td>
<td>-2.30</td>
<td>4.39</td>
<td>2.59</td>
</tr>
</tbody>
</table>

Table 6.27: Results of Conducting the Two-Tailed Independent-Samples Significance Test (t-Test) on Complex Variables: Effectiveness, Efficiency, and Satisfaction

<table>
<thead>
<tr>
<th>df=58</th>
<th>Complex Variables</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
<td>Satisfaction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>2.16</td>
<td>3.07</td>
<td>4.51</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.28: Results of Conducting the Two-Tailed Independent-Samples Significance Test (t-Test) on Usable-Security Level

<table>
<thead>
<tr>
<th>df=58</th>
<th>Usable-Security</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>4.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.29: Results of Conducting the One-Way Randomized ANOVA Test (F-Test) on Simple Variables: task completion (Completion); number of errors (Errors); sign-up time; login time; degree of positive affect (Happiness); ease of use (Ease); and degree of negative affect (Clarity)

<table>
<thead>
<tr>
<th>df=58</th>
<th>Simple Variables</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Completion</td>
<td>0.34</td>
<td>6.76</td>
<td>19.02</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
<td>8.52</td>
<td>5.22</td>
<td>6.67</td>
</tr>
<tr>
<td></td>
<td>Sign-up</td>
<td></td>
<td></td>
<td>19.64</td>
</tr>
<tr>
<td></td>
<td>Login</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Happiness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ease</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clarity</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.30: Results of Conducting the One-Way Randomized ANOVA Test (F-Test) on Complex Variables: Effectiveness, Efficiency, and Satisfaction

<table>
<thead>
<tr>
<th>df=58</th>
<th>Complex Variables</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>4.62</td>
<td>9.34</td>
<td>20.21</td>
</tr>
</tbody>
</table>

One-Way Randomized Analysis of Variance (F-Test)

The result of the above Two-Tailed Independent-Samples Significance Test (t-Test) shows that the results of our research experiment provide enough statistical evidence that there is significant difference between the CBAA and TBAA in terms of Usable-Security level. However, we decided to conduct another parametric statistical analysis in order to doubly confirm that there is enough significant difference between the two approaches and the null hypothesis should be rejected (see Tables 6.29, 6.30, and 6.31).

Similar to the results of the previous t-Test, the results of F-scores on the dependent variables (both simple and complex) indicate that only one dependent variable (Task Completion, Completion) does not statistically support rejecting the null hypothesis because most of the participants completed their sessions successfully. On the other hand, F-scores of the rest of the variables support rejecting the null hypothesis, as all of their F-scores are beyond the critical value of 4.000 [60]. Therefore, the overall result
Table 6.31: Results of Conducting the One-Way Randomized ANOVA Test (F-Test) on Usable-Security Level

<table>
<thead>
<tr>
<th>F-Test [F(1, 58), α=0.05, Critical Value=4.000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=58</td>
</tr>
<tr>
<td>F</td>
</tr>
</tbody>
</table>

of the F-Test on Usable-Security is: [F(1,58)=20.40, p <0.05].

Same as the results of the previous t-Test, the null hypothesis assuming that there is no difference in the usable-security component and individual measures between the choice-based security approach and the traditional-based security approach when used by end-users, is rejected. The alternative hypotheses are supported.

Comparing Means

After using the two previous statistical significance tests to evaluate the variances that can be explained by the variables and the variances that cannot [73], we discovered that the only variable that cannot be explained is the Task Completion variable, Completion. The remainder of the variables can be explained. Therefore, the means of the explainable dependent variables can be compared and used to compare between the CBAA and the TBAA approaches. The following sections compare between the means of: the simple dependent variables, complex dependent variables, and overall Usable-Security level (which is the most important to be compared between the two approaches).

Means of Simple Variables.

Figure 6.9 shows the means of each simple dependent variable. It worthwhile to note that the “Task Completion, Completion” variable should not be considered. As based on the conducted statistical significance tests, it does not provide enough evidence (no
statistically significant difference based on both t-Test and F-Test: \[ t(58)=0.59, p < 0.05 \]
and \[ F(1,58)=0.34, p < 0.05 \], respectively. The remainder of the simple dependent variables were considered.

The means of the Number of Errors variable show that the CBAA received 12 percent, where the TBAA received 45 percent. This indicates that the CBAA gets lower error rate (users made fewer errors) than the TBAA.

The means of the Sign-up Time variable show that the CBAA gets 23 percent, while the TBAA gets 32 percent. This infers that the CBAA gets lower sign-up time (users spent less time to sign-up) than the TBAA. Similarly, the means of Login Time variable show that the CBAA gets 44 percent, while the TBAA gets 53 percent. This infers that the CBAA gets lower login time (users spent less time to login) than the TBAA.

The means of the Degree of Positive Affect (Happiness) variable indicates that the participants who were randomly assigned to work on the CBAA are happier than those who worked on the TBAA, because the CBAA gets 86 percent as average happiness, where the TBAA gets 63 percent. Similarly, the means of the Ease of Use variable support the indication that the CBAA seems to be easier for usage than the TBAA because the mean of the Ease of Use variable for the CBAA is 88 percent, which is higher than the mean of the Ease of Use variable for TBAA, which is 63 percent. Finally, the means of the Degree of Clarity variable show that the CBAA seems to be clearer to the users than the TBAA because the mean of the CBAA is 90 percent, where the TBAA is 63 percent.

Figure 6.9 results support some alternative hypotheses \( H_3, H_4, \) and \( H_5 \); however, the alternative hypothesis \( H_1 \) is not supported because the statistical significance analysis does not show enough significant difference between the two approaches on the Task Completion variable. Moreover, the alternative hypothesis \( H_2 \) should not be
Figure 6.9: The Means of the Simple Dependent Variables for both Approaches: the CBAA and the TBAA
supported, and should be rejected. As the conducted significance tests show that there is significant difference between the two approaches on the Sign-up Time variable. Hence, we can rely on the means of such variable to indicate that the CBAA participants have spent less time to sign-up.

**Means of Complex Variables.**

Figure 6.10 shows the means of each complex dependent variable. The complex variables can be considered because, according to the conducted statistical significance tests, they demonstrate significant differences between the means of the two approaches (CBAA and TBAA). As seen in Figure 6.10, the three variables indicate that the CBAA receives higher means than the TBAA.

The mean of *Effectiveness* variable for the CBAA is 93 percent, where the TBAA is 83 percent. Moreover, the mean of *Efficiency* variable for the CBAA is 67 percent, while the TBAA is 58 percent. In addition, the mean of *Satisfaction* variable for the CBAA is 88 percent, and for the TBAA is 64 percent.

The means of Figure 6.10 support the alternative hypotheses ($H_6$, $H_7$, $H_8$, and $H_9$), as the CBAA receives higher means in all of the three dependent variables: Effectiveness, Efficiency, and Satisfaction. Hence, the CBAA is better than the TBAA in terms of the three complex variables.

**Means of Usable-Security.**

Comparing the means of Usable-Security between the CBAA and the TBAA is the core of this experiment, and the primary objective is to figure out whether the two approaches have the same usable-security level or if there is a difference between such approaches. Consequently, this section investigates whether the null hypothesis ($H_0$) can be rejected or not. Figure 6.11 displays the means of both approaches. The mean
Figure 6.10: The Means of the Complex Dependent Variables for both Approaches: the CBAA and the TBAA
Figure 6.11: The Means of Usable-Security for both Approaches: the CBAA and the TBAA

of Usable-Security for the CBAA is 0.83 out of 1, while the mean for the TBAA is 0.68 out of 1.

Based on the conducted statistical significance tests (t-Test and F-Test), there is a strong significant difference between the CBAA and the TBAA on usable-security, as the results of both t-Test and F-Test are $[t(58)=4.54, p < 0.05]$, $[F(1, 58)=20.40, p < 0.05]$ respectively. By comparing the means of Usable-Security between the CBAA and the TBAA, it is obvious that the mean of the CBAA is higher than the mean of the TBAA. Hence, it can be inferred that the CBAA probably provide better usable-security than the TBAA.
Correlation Analysis

As part of statistical analysis for our experiment, we investigated the relationships between the variables through conducting correlation coefficient tests [73]. Two levels of correlation analyses have been conducted: an overall correlation analysis and per approach correlation analysis. The following sections describe both levels of the correlation analyses, where the value of Pearson’s $r$ ranges between perfect negative relationship (Pearson’s $r = -1.00$) and perfect positive relationship (Pearson’s $r = 1.00$). It worthwhile to note that when Pearson’s $r$ value is zero (Pearson’s $r = 0.00$) between two particular variables, this means that there is no linear relationship between them and the two variables are independent from one another. Logically, each variable should be in a perfect positive relationship with itself (Pearson’s $r$ value is 1.00). Hence, such values should be ignored during the correlation analysis.

Pearson’s $r$ magnitude values are categorized as strong ($\pm 0.70 – 1.00$), moderate ($\pm 0.30 – 0.69$), none (0.00) or weak ($\pm 0.00 – 0.29$) [60].

**Overall Correlation Analysis.**

This level of correlation test combines both approaches (the CBAA and the TBAA) together in one correlation analysis study in order to identify relationships between the dependent variables, and between the dependent variables and Usable-Security (see Tables 6.32 and 6.33).

Table 6.32 displays the correlation coefficient values (Pearson’s $r$ values) between the variables. The Task Completion variable shows neither strong relationship with the other dependent variables nor with Usable-Security. The Number of Errors variable reveals strongly negative significant relationships with the Effectiveness variable
Table 6.32: Pearson’s $r$ Values of the Overall Correlation Analysis on Simple Variables: task completion (Completion), number of errors (Errors), sign-up time, login time, degree of positive affect (Happiness), ease of use (Ease), and degree of negative affect (Clarity). Note: Usa-Sec stands for Usable-Security.

<table>
<thead>
<tr>
<th>Simple Variables</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion</td>
<td>1.00</td>
<td>-0.29</td>
<td>0.02</td>
</tr>
<tr>
<td>Errors</td>
<td>-0.29</td>
<td>1.00</td>
<td>0.63</td>
</tr>
<tr>
<td>Sign-up</td>
<td>0.02</td>
<td>0.63</td>
<td>1.00</td>
</tr>
<tr>
<td>Login</td>
<td>-0.01</td>
<td>0.13</td>
<td>0.34</td>
</tr>
<tr>
<td>Happiness</td>
<td>0.13</td>
<td>-0.52</td>
<td>-0.45</td>
</tr>
<tr>
<td>Ease</td>
<td>-0.04</td>
<td>-0.31</td>
<td>-0.42</td>
</tr>
<tr>
<td>Clarity</td>
<td>-0.03</td>
<td>-0.43</td>
<td>-0.51</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.69</td>
<td>-0.87</td>
<td>-0.52</td>
</tr>
<tr>
<td>Efficiency</td>
<td>-0.01</td>
<td>-0.46</td>
<td>-0.81</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.01</td>
<td>-0.47</td>
<td>-0.52</td>
</tr>
<tr>
<td>Usa-Sec</td>
<td>0.01</td>
<td>-0.76</td>
<td>-0.75</td>
</tr>
</tbody>
</table>

(Pearson’s $r = -0.87$) and with Usable-Security (Pearson’s $r = -0.76$). It does not show strongly positive relationships with the other variables.

The Sign-up Time variable shows strongly negative significant relationships with the Efficiency variable (Pearson’s $r = -0.81$) and with Usable-Security (Pearson’s $r = -0.75$). It does not reveal strongly positive relationships. The Login Time variable shows only one strongly negative significant relationship with the Effectiveness variable (Pearson’s $r = -0.83$), but does not show any other strong relationships (neither positive nor negative).

The Degree of Positive Affect variable, *Happiness*, does not show any strongly negative significant relationship. However, it shows strongly positive relationships with the Degree of Negative Affect variable (Clarity) (Pearson’s $r = 0.73$) and the Satisfaction variable (Pearson’s $r = 0.88$), as well as Usable-Security (Pearson’s $r = 0.79$). The Ease of Use variable only shows one strongly positive significant relationship with the Satisfaction variable (Pearson’s $r = 0.84$), but does not show any other strong...
Table 6.33: Pearson’s $r$ values of the Overall Correlation Analysis on Complex Variables (Effectiveness, Efficiency, and Satisfaction), and Usable-Security

<table>
<thead>
<tr>
<th>N=60</th>
<th>Complex Variables</th>
<th>Usable-Security</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>1.00</td>
<td>0.36</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.36</td>
<td>1.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.39</td>
<td>0.44</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.74</td>
<td>0.70</td>
</tr>
</tbody>
</table>

significant relationships. The Degree of Negative Affect (Clarity) shows three strongly positive significant relationships with the Degree of Positive Affect variable (Happiness) (Pearson’s $r = 0.73$) and the Satisfaction variable (Pearson’s $r = 0.95$), as well as Usable-Security (Pearson’s $r = 0.81$).

As shown in Table 6.33, the Effectiveness variable shows one strongly negative significant relationship with the Number of Errors variable (Pearson’s $r = -0.87$). Conversely, it shows one strongly positive significant relationship with Usable-Security (Pearson’s $r = 0.74$). The Efficiency shows two strongly negative significant relationships with the Sign-up Time variable (Pearson’s $r = -0.81$) and the Login Time variable (Pearson’s $r = -0.83$). However, it shows one strongly positive significant relationship with Usable-Security (Pearson’s $r = 0.70$). The Satisfaction variable only shows four strongly positive significant relationships with the Degree of Positive Affect (Happiness) (Pearson’s $r = 0.88$), the Ease of Use variable (Pearson’s $r = 0.84$), and the Degree of Negative Affect (Clarity) (Pearson’s $r = 0.95$), as well as Usable-Security (Pearson’s $r = 0.86$).

To conclude the overall correlation analysis, Usable-Security shows two strongly negative significant relationships with the following variables: the Number of Errors and the Sign-up Time (Pearson’s $r$ values are: -0.76 and -0.75, respectively). Conversely, it shows five strongly positive significant relationships with the following variables:
Table 6.34: Pearson’s $r$ Values of the CBAA Correlation Analysis on Simple Variables: task completion (Completion), number of errors (Errors), sign-up time, login time, degree of positive affect (Happiness), ease of use (Ease), and degree of negative affect (Clarity). Note: Usa-Sec stands for Usable-Security.

<table>
<thead>
<tr>
<th>Simple Variables</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Completion</td>
<td>Errors</td>
<td>Sign-up</td>
</tr>
<tr>
<td>N=30</td>
<td>Completion</td>
<td>Errors</td>
<td>Sign-up</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>-0.34</td>
<td>-0.04</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
<td>1.00</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>Sign-up</td>
<td>-0.04</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>Login</td>
<td>-0.01</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>Happiness</td>
<td>0.32</td>
<td>-0.69</td>
</tr>
<tr>
<td></td>
<td>Ease</td>
<td>-0.12</td>
<td>-0.03</td>
</tr>
<tr>
<td></td>
<td>Clarity</td>
<td>-0.08</td>
<td>-0.18</td>
</tr>
<tr>
<td></td>
<td>Effectiveness</td>
<td>0.70</td>
<td>-0.90</td>
</tr>
<tr>
<td></td>
<td>Efficiency</td>
<td>0.03</td>
<td>-0.34</td>
</tr>
<tr>
<td></td>
<td>Satisfaction</td>
<td>0.05</td>
<td>-0.38</td>
</tr>
<tr>
<td></td>
<td>Usa-Sec</td>
<td>0.45</td>
<td>-0.83</td>
</tr>
</tbody>
</table>

Degree of Positive Affect variable (Happiness), Degree of Negative Affect variable (Clarity), Effectiveness, Efficiency, and Satisfaction. Pearson’s $r$ values are: 0.79, 0.81, 0.74, 0.70, and 0.86, respectively. Hence, it can be statistically inferred that the higher usable-security level a system achieves, the lower the sign-up and login times the end-users experience. In contrast, the higher usable-security level a system achieves, the higher happiness and satisfaction levels the end-users feel, and the better clarity, effectiveness, and efficiency the systems demonstrate.

**CBAA Correlation Analysis**

This analysis only considers the CBAA approach. Throughout this correlation coefficient test, we investigate relationships between the dependent variables; and between the dependent variables and Usable-Security for the CBAA only (see Tables 6.34 and 6.35).
According to the Pearson’s $r$ values on Table 6.34, the Task Completion variable shows neither a strong relationship with the other dependent variables, nor with Usable-Security. The Number of Errors variable shows a strongly negative significant relationship with the Effectiveness variable (Pearson’s $r = -0.90$). In addition, it shows another strongly negative significant relationship with Usable-Security (Pearson’s $r = -0.83$). However, the Number of Errors variable does not show any strongly positive significant relationships.

The Sign-up Time variable shows a strongly negative significant relationship with the Efficiency variable (Pearson’s $r = -0.73$), as well as another strongly negative significant relationship with Usable-Security (Pearson’s $r = -0.76$). Nevertheless, the Sign-up Time variable does not reveal strongly positive significant relationships. The Login Time variable shows only one strongly negative significant relationship with the Efficiency variable (Pearson’s $r = -0.83$), and does not show strongly positive significant relationships.

The Degree of Positive Affect variable (Happiness) shows a strongly positive significant relationship with the Satisfaction variable (Pearson’s $r = 0.82$). In addition, it shows a strongly positive significant relationship with Usable-Security (Pearson’s $r = 0.87$). In contrast, the Degree of Positive Affect variable (Happiness) does not reveal strongly negative significant relationships. The Ease of Use variable shows only one strongly positive significant relationship with the Satisfaction variable (Pearson’s $r = 0.72$), and does not show strongly negative relationships. Similarly, the Degree of Negative Affect variable (Clarity) shows only one strongly positive significant relationship with the Satisfaction variable (Pearson’s $r = 0.83$), and does not show strongly negative relationships.
Table 6.35: Pearson’s $r$ Values of the CBAA Correlation Analysis on Complex Variables (Effectiveness, Efficiency, and Satisfaction), and Usable-Security

<table>
<thead>
<tr>
<th>N=30</th>
<th>Complex Variables</th>
<th>Usable-Security</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>1.00</td>
<td>0.28</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.28</td>
<td>1.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.34</td>
<td>0.03</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.85</td>
<td>0.53</td>
</tr>
</tbody>
</table>

As seen in Table 6.34, the Effectiveness variable shows one strongly negative significant relationship with the Number of Errors variable (Pearson’s $r = -0.90$). In return, it shows one strongly positive significant relationship with Usable-Security (Pearson’s $r = 0.85$). The Efficiency variable shows two strongly negative relationships with the following variables: Sign-up Time and Login Time (Pearson’s $r$ values are: -0.73 and -0.83, respectively), but does not show strongly positive relationships. The Satisfaction variable shows three strongly positive significant relationships with the following variables: the Degree of Positive Affect (Happiness), Ease of Use, and Degree of Negative Affect (Clarity), (Pearson’s $r$ values are: 0.82, 0.72, and 0.83, respectively). Merely, the Satisfaction variable does not show strongly negative relationships.

To summaries the CBAA correlation analysis, Usable-Security shows two strongly negative significant relationships with the following variables: the Number of Errors and the Sign-up Time (Pearson’s $r$ values are: -0.83 and -0.76, respectively). In return, Usable-Security shows two strongly positive significant relationships with: the Degree of Positive Affect (Happiness) and Effectiveness\(^1\) (Pearson’s $r$ values are: 0.87 and 0.85, respectively).

\(^1\)The result of the correlation analysis between Usable-Security and Effectiveness is displayed in Table 6.35
Table 6.36: Pearson’s *r* values of the TBAA correlation analysis on simple variables: task completion (Completion), number of errors (Errors), sign-up time, login time, degree of positive affect (Happiness), ease of use (Ease), and degree of negative affect (Clarity). Note: Usa-Sec stands for Usable-Security.

<table>
<thead>
<tr>
<th>N=60</th>
<th>Simple Variables</th>
<th>Effectiveness</th>
<th>Efficiency</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Completion</td>
<td>Errors</td>
<td>Sign-up</td>
</tr>
<tr>
<td>Completion</td>
<td>1.00</td>
<td>-0.24</td>
<td>0.09</td>
<td>0.03</td>
</tr>
<tr>
<td>Errors</td>
<td>-0.24</td>
<td>1.00</td>
<td>0.56</td>
<td>0.08</td>
</tr>
<tr>
<td>Sign-up</td>
<td>0.09</td>
<td>0.56</td>
<td>1.00</td>
<td>0.30</td>
</tr>
<tr>
<td>Login</td>
<td>0.03</td>
<td>0.08</td>
<td>0.30</td>
<td>1.00</td>
</tr>
<tr>
<td>Happiness</td>
<td>0.01</td>
<td>-0.26</td>
<td>-0.26</td>
<td>-0.08</td>
</tr>
<tr>
<td>Ease</td>
<td>-0.05</td>
<td>-0.32</td>
<td>-0.40</td>
<td>-0.19</td>
</tr>
<tr>
<td>Clarity</td>
<td>-0.08</td>
<td>-0.39</td>
<td>-0.49</td>
<td>-0.23</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.70</td>
<td>-0.83</td>
<td>-0.46</td>
<td>-0.03</td>
</tr>
<tr>
<td>Efficiency</td>
<td>-0.07</td>
<td>-0.40</td>
<td>-0.81</td>
<td>-0.80</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>-0.06</td>
<td>-0.37</td>
<td>-0.45</td>
<td>-0.20</td>
</tr>
<tr>
<td>Usa-Sec</td>
<td>0.23</td>
<td>-0.69</td>
<td>-0.71</td>
<td>-0.39</td>
</tr>
</tbody>
</table>

**TBAA Correlation Analysis**

This analysis only considers the TBAA approach. Throughout this correlation coefficient test, we investigated relationships between the dependent variables, and between the dependent variables and Usable-Security for the TBAA only (see Tables 6.36 and 6.37).

Pearson’s *r* values on Table 6.36 provide various identifications for relationships. The Task Completion variable shows a strongly positive significant relationship with the Effectiveness variable (Pearson’s *r* = 0.70), and does not show strongly negative relationships with the other variables. The Number of Errors variable shows only one strongly negative significant relationship with the Effectiveness variable (Pearson’s *r* = -0.83), but does not show strongly positive significant relationships.

The Sign-up Time variable shows two strongly negative significant relationships with Efficiency variable and Usable-Security (Pearson’s *r* values: -0.81 and -0.71,
respectively). However, it does not show strongly positive relationships. The Login Time variable shows only one negative significant relationship with the Efficiency variable (Pearson’s $r = -0.80$), but it does not show strongly positive relationships.

The Degree of Positive Affect variable, *Happiness*, shows three strongly positive significant relationships with the Ease of Use, the Degree of Negative Affect (*Clarity*), and Satisfaction (Pearson’s $r$ values are: 0.70, 0.71, and 0.86, respectively). The Ease of Use variable shows four strongly positive significant relationships with the Degree of Positive Affect (*Happiness*), the Degree of Negative Affect (*Clarity*), Satisfaction, and Usable-Security. Pearson’s $r$ values are: 0.70, 0.71, 0.86, and 0.74, respectively. Nevertheless, the Ease of use variable does not show strongly negative relationships. The Degree of Negative Affect variable (*Clarity*) shows four strongly positive significant relationships with Degree of Positive Affect (*Happiness*), the Ease of Use, Satisfaction, and Usable-Security. Pearson’s $r$ values are: 0.71, 0.71, 0.95, and 0.83, respectively. The Degree of Negative Affect (*Clarity*) shows no strongly negative relationships.

Table 6.37 presents correlation values between the complex dependent variables and Usable-Security. The Effectiveness variable shows two strong significant relationships, one negative and one positive. The strongly negative significant relationship is with the Number of Errors variable (Pearson’s $r = -0.83$), while the strongly positive relationship is with the Task Completion variable (Pearson’s $r = 0.70$). The Efficiency variable shows two strongly negative significant relationships with the Sign-up Time and the Login Time variables. Pearson’s $r$ values are: -0.81 and -0.80, respectively. The variable does not show strongly positive relationships. The Satisfaction variable shows four strongly positive significant relationships with the Degree of Positive Affect (*Happiness*), the Ease of Use, the Degree of Negative Affect, and Usable-Security. Pearson’s $r$ values are: 0.86, 0.86, 0.95, and 0.85, respectively.
Table 6.37: Pearson’s $r$ values of the TBAA correlation analysis on complex variables (Effectiveness, Efficiency, and Satisfaction), and Usable-Security

<table>
<thead>
<tr>
<th>N=30</th>
<th>Complex Variables</th>
<th>Usable-Security</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>1.00</td>
<td>0.31</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.31</td>
<td>1.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.31</td>
<td>0.41</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.69</td>
<td>0.68</td>
</tr>
</tbody>
</table>

To conclude the TBAA correlation analysis, Usable-Security shows four strong relationships: one negative and three positive. The strongly negative significant relationship of Usable-Security is with the Sign-up Time variable (Pearson’s $r = -0.71$). On the other hand, the three strongly positive significant relationships are with the Ease of Use, the Degree of Negative Affect, and Satisfaction. Pearson’s $r$ values are: 0.74, 0.83, and 0.85, respectively.

**Correlation Analysis Summary**

From the above three correlation analyses sections, we calculated the mean of each relationship based on the three Pearson’s $r$ values from the three correlation analyses: overall, CBBA, and TBAA. The result of calculating the mean is considered for concluding the significantly correlation relationships between Usable-Security and the dependent variables. Tables 6.38 and 6.39 display the resulted means of Pearson’s $r$ values for the relationships. However, we only focused on the usable-security correlations.

As seen in Tables 6.38 and 6.39, Usable-Security shows five strong significant relationships: two negatives and three positives. The two strongly negative significant relationships are with the Number of Errors and the Sign-up Time variables. The mean Pearson’s $r$ values are: -0.76 and -0.74, respectively. While the three strongly
Table 6.38: Pearson’s $r$ values of the average correlation analysis on simple variables: task completion (Completion), number of errors (Errors), sign-up time, login time, degree of positive affect (Happiness), ease of use (Ease), and degree of negative affect (Clarity). Note: Usa-Sec stands for Usable-Security.

<table>
<thead>
<tr>
<th></th>
<th>Simple Variables</th>
<th></th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Effectiveness</td>
<td>Efficiency</td>
<td></td>
</tr>
<tr>
<td>Completion</td>
<td>1.00</td>
<td>-0.29</td>
<td>0.02</td>
</tr>
<tr>
<td>Errors</td>
<td>-0.29</td>
<td>1.00</td>
<td>0.62</td>
</tr>
<tr>
<td>Sign-up</td>
<td>0.02</td>
<td>0.62</td>
<td>1.00</td>
</tr>
<tr>
<td>Login</td>
<td>0.00</td>
<td>0.05</td>
<td>0.28</td>
</tr>
<tr>
<td>Happiness</td>
<td>0.15</td>
<td>-0.49</td>
<td>-0.43</td>
</tr>
<tr>
<td>Ease</td>
<td>-0.07</td>
<td>-0.22</td>
<td>-0.34</td>
</tr>
<tr>
<td>Clarity</td>
<td>-0.07</td>
<td>-0.33</td>
<td>-0.40</td>
</tr>
</tbody>
</table>

Table 6.39: Pearson’s $r$ values of the average correlation analysis on complex variables (Effectiveness, Efficiency, and Satisfaction), and Usable-Security

<table>
<thead>
<tr>
<th></th>
<th>Complex Variables</th>
<th>Usable-Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness</td>
<td>1.00</td>
<td>0.32</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0.32</td>
<td>1.00</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>0.35</td>
<td>0.29</td>
</tr>
<tr>
<td>Usable-Security</td>
<td>0.76</td>
<td>0.64</td>
</tr>
</tbody>
</table>
positive significant relationships are with the Degree of Positive Affect, Effectiveness, and Satisfaction. The mean Pearson’s $r$ values are: 0.77, 0.76, and 0.79, respectively.
6.5 Conclusion

Throughout this chapter, our proposed framework (the USEF) has been evaluated in various ways. A heuristic evaluation was conducted on the framework in Section 6.2. The overall result of the evaluation is encouraging, as the final result indicates that the framework attained 4 out of 5 (80 percent) when evaluated by security, usability, software engineers, and usable-security experts. Moreover, we have proposed a Choice-Based Authentication Approach (CBAA), and have applied the approach through developing an authentication system based on the proposed framework as a demonstration in Section 6.3. After that, in Section 6.4, the CBAA is compared against the Traditional-Based Authentication Approach (TBAA) that represents the typical authentication systems’ development processes without adopting the USEF. The results of the conducted parametric statistical analysis from the comparative experiment (t-Test and F-Test) indicate that there are enough significant differences between the two approaches (i.e.: CBAA and TBAA). Hence, according to the results of the analysis, incorporating the USEF within the process of developing software systems when aligning security and usability is an important matter in helping usable-security enhancement.
Chapter 7

Conclusion and Future Directions

7.1 Summary

Throughout our research, we have focused on integrating two conflicting software quality attributes that are important for most software products, namely: security and usability. The goal of this research work is to design a framework that helps enhance the relationship between the two attributes so they can interact synergistically.

In Chapter 1, towards reaching our goal, this research stated four problems to be addressed. First, the nature of the relationship between security and usability is not clear enough. Second, there is a lack of research work on integrating security and usability during the requirements engineering phase. Third, most of the proposed solutions during the design phase experience weaknesses. Fourth, both security and usability are considered as subjective quality attributes.

During our literature review in Chapter 2, we realized that in order to address the above problems that limit integrating security and usability, any proposed solution must consider the Software Development Lifecycle (SDLC) and initiate the work from the
initial steps of the development process (i.e: from the requirements engineering phase). In addition, such solution should also be sequential following the bottom-up approach. Note that each part of the sequence should be flexible enough to work individually and independently from the other parts when needed.

In Chapter 3, the first component of our framework provides an assessment methodology that helps gauge the desired balance between security and usability during the requirements engineering phase. The assessment process is based on well-known techniques adopted from the decision science field. After proposing, prototyping, testing, and continually enhancing the assessment methodology, we moved to the next phase of the SDLC, namely: the design phase.

During the design phase, Chapter 4, the second component of our framework provides a set of usable-security design guidelines that is built based on the work of the first component of the requirements engineering phase. It provides systematic guidance for linking the usability, security, and usable-security requirements to appropriate design tools (i.e. principles, practices, patterns, and techniques) that can be used during the architectural design stage. The trade-off between the selected design tools is then resolved. However, it can be used individually and independently from the first component. Moving forward, the construction phase has not been included within the scope of our framework because it is an implementation process that applies the work of its preceding phases. Moreover, the construction phase shows enough stability and ability to meet the needs of its preceding phases.

Consequently, in Chapter 5, the third component of our framework provides a summative usable-security measuring matrix that serves as a final checkpoint for applying the AFUS and the guidelines in order to validate the resolved trade-offs between security and usability.
Finally, Chapter 6 provides a human-based experiment on a proposed authentication approach built based on the USEF as a demonstration. In addition, the chapter provides a heuristic evaluation that overall evaluates the whole framework (the components and their interaction).

### 7.2 Limitations

The usable-security engineering framework provides an adequate foundation for providing solutions for integrating security and usability. However, like any research work, our journey towards making the two attributes synergistic faced some limitations. The fact that security and usability have conflicting goals makes any work to integrate them complicated, which is by itself a major limitation. This might explain the reason for having limited research work in this field, which is another limitation.

Dealing with humans is bothersome, as dealing with human beings means dealing with their minds and moods. This fact can affect the results of any human-related research work; therefore, we struggled during performing our experiments and studies that incorporate human-beings as subjects. One way of struggling is that in order to conduct a human-related experiment, there are extra efforts and processes to be done such as: seeking for funding, allocating and preparing a usability-testing lab within the campus area, gaining the permission from the Institutional Review Board-IRB, and recruiting participants. The USEF has been proposed and tested against small and medium-size scenarios and use cases. Despite the promising results of developing a relatively medium-size authentication system that demonstrates the application of the framework, it needs to be used for developing large-scale software projects that importantly consider integrating security and usability. The USEF is a novel systematic
method for developing software products with balanced integration between security and usability; therefore, it is currently still unstable and has not reached the targeted degree of maturity to be used in large-size software projects. This fact makes it limited, to an extent, to only small and medium size software projects rather than large projects.

Adopting the utility functions and the decision tree as well-known key tools from the field of decision science to the field of computer sciences is another limitation. Dealing with such tools within the scope of the SDLC for software development requires providing adequate training to the development team, especially requirements engineers.

The security and usability literature is mature enough in documenting and specifying the optimal ways of incorporating design tools (i.e: principles, patterns, techniques, and best practices). despite the promising work of the USEF, the process of associating security, usability, and usable-security requirements with their appropriate design tools during the design phase needs significant efforts and security-usability expertise that sufficiently enable applying the correct set of such tools properly in real software development cases.

7.3 Future Directions

Our research to design a framework that enhances the balancing of integrating the two conflicting software quality attributes, security and usability, has partially incorporated various disciplines of science as security engineering, usability engineering, software engineering, ergonomics engineering, systems engineering, and psychology are all participated in designing the framework.

In the future, we will continue working towards enhancing the framework through involving specialists from each of the incorporated disciplines to provide careful
enhancement of each part. Of course, such enhancement process needs long-term planning, because the enhancement should be divided into milestones. Each milestone will focus on one area of potential enhancement with consideration for the aspects that may affect the other parts of the framework.

Although we provided high-level fidelity prototypes for processing the components of the framework, we will have them executed (when they become robust and mature adequately) in order to make them dynamically intractable and flexibly adaptable.

In addition, we will work to discover if it is possible to determine an optimal balance between security and usability (i.e: the optimal usable-security level). We believe that this might be achieved by adopting the trade-off charts of queuing theory. The trade-off chart is an approach to deal with the problem of applying a queuing model in order to make the decision of the optimal capacity between the system’s cost and the waiting cost. As the system’s cost increases, the waiting cost typically decreases. Similarly, the optimal balance of the trade-off between security and usability might be achieved by adopting the same trade-off model that is used to determine optimal capacity in the queuing theory. This research can be conducted after applying the framework on real several systems, and the resulted values of measuring security, usability, and usable-security are available. Based on the resulted values, the trade-off chart is created and exponential trend-lines of both security and usability are drawn. Finally, the intersection between the two trend-lines could be considered as the optimal balance of the trade-off between security and usability. Appendix D provides a depiction on the potential application of the trade-off chart on security and usability.
Bibliography


Appendix A

List of Contributions

The following is the list of contributions that have been published during the course of this Ph.D. research:


253

Appendix B

Applying the AFUS

This appendix presents the processes of applying the AFUS for developing the CBAA (The USEF framework’s demonstration), as the AFUS has been proposed in Chapter 3 and applied on the CBAA in Section 6.3.1.

B.1 Requirements Filtering and Merging
Figure B.1: The Security Requirements. The security requirements after being filtered and rated based on the security and usability properties' importance rating guide of Chapter 3 (Table 3.1).

<table>
<thead>
<tr>
<th>Code</th>
<th>Requirement</th>
<th>CI</th>
<th>NI</th>
<th>AI</th>
<th>SecR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The system shall be available for use 24 hours per day, 365 days per year</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>3.33333</td>
</tr>
<tr>
<td>2</td>
<td>The system shall only allow authorized users to access authorized data and resources</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>4.66667</td>
</tr>
<tr>
<td>3</td>
<td>The system shall only allow authorized users to alter authorized data and only in a proper way</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>4.66667</td>
</tr>
<tr>
<td>4</td>
<td>The system shall hide the details of its construction from the user.</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4.88889</td>
</tr>
<tr>
<td>5</td>
<td>The system shall provide a separate login interface</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3.33333</td>
</tr>
<tr>
<td>6</td>
<td>The system shall show all authentication methods on the login interface</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>3.66667</td>
</tr>
<tr>
<td>7</td>
<td>The system shall meet the NIST 000-Series</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>The system shall not store users' private information</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4.66667</td>
</tr>
<tr>
<td>9</td>
<td>The system shall make the blind guessing attack difficult to occur</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33333</td>
</tr>
<tr>
<td>10</td>
<td>The system shall make the focused guessing attack difficult to occur</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33333</td>
</tr>
<tr>
<td>11</td>
<td>The system shall make the observation attack difficult to occur</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3.33333</td>
</tr>
<tr>
<td>12</td>
<td>The system shall provide a separate sign-up interface</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.66667</td>
</tr>
<tr>
<td>13</td>
<td>The system shall provide least granting authority</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>66</td>
<td>46</td>
<td>31</td>
<td>48.33333</td>
</tr>
</tbody>
</table>

\[ SEC = \sum_r SecR_r \]
\[ SH_c = \frac{\sum_r CI_r}{SEC} \]
\[ SH_n = \frac{\sum_r NI_r}{SEC} \]
\[ SH_u = \frac{\sum_r AI_r}{SEC} \]
\[ SecR_r = \frac{CI_r + NI_r + AI_r}{3} \]

\[ ShC_{1} = 1.33332 \]
\[ ShH_{1} = 0.99381 \]
\[ ShU_{1} = 0.64138 \]
\[ SEC_{1} = 48.33333 \]
Figure B.2: The Usability Requirements. The usability requirements after being filtered and rated based on the security and usability properties importance rating guide of Chapter 3 (Table 3.1).

<table>
<thead>
<tr>
<th>Code (i)</th>
<th>Requirement</th>
<th>E1</th>
<th>F1</th>
<th>SI</th>
<th>UsaR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The system shall be effective by making the users able to complete all the required tasks</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6.44667</td>
</tr>
<tr>
<td>2</td>
<td>The system shall help the users to avoid making mistakes by providing clear steps to be followed</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>3.66667</td>
</tr>
<tr>
<td>3</td>
<td>The system shall make the users satisfied and want to use it by adopting user-friendly interfaces</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>5.33333</td>
</tr>
<tr>
<td>4</td>
<td>The system shall allow users to complete the required tasks in an acceptable amount of time</td>
<td>3</td>
<td>9</td>
<td>7</td>
<td>8.33333</td>
</tr>
<tr>
<td>5</td>
<td>The system shall be as universally accessible as possible by providing at least three methods of authentication</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>The system shall not increase the users' cognitive load through providing consistent steps and interfaces</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>The system shall provide an acceptable sign-up time that doesn't exceed the standard average sign-up time</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>5.33333</td>
</tr>
<tr>
<td>8</td>
<td>The system shall provide an acceptable login time that doesn't exceed the standard average login time</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>5.33333</td>
</tr>
<tr>
<td>9</td>
<td>The system shall be easy to use by making the users self-sufficient and not need to guidance to use it</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>The system shall not confuse the users by avoiding duplication, double visual, implicit negatives, clashing alternatives</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>The system shall provide complete freedom to the users to choose whichever authentication method they prefer</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>The system shall provide comfortable ways for the users to perform tasks</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

\[
USA = \sum_{j=1}^{UR} \frac{U_{ja}F_j}{USA} \\
SH_e = \frac{\sum_{j=1}^{UR} E_{lj}}{USA} \\
SH_f = \frac{\sum_{j=1}^{UR} F_{lj}}{USA} \\
SH_s = \frac{\sum_{j=1}^{UR} S_{lj}}{USA} \\
UsaR = \frac{E_{lj} + F_{lj} + S_{lj}}{3}
\]

\[
SHe = SHf = SHs = 0.92831
\]

257
Figure B.3: The Initial Usable-Security Requirements. The initial usable — security requirements after being filtered and rated based on the initial usable — security importance rating guide of Chapter 3 (Table 3.2).
Figure B.4: The Merged Usable-Security Requirements. The merged usable–security requirements after being derived from aligning security and usability requirements in Chapter 3.

<table>
<thead>
<tr>
<th>Code (i)</th>
<th>Requirement</th>
<th>CSR</th>
<th>$\sum_{SecR}^i$</th>
<th>CUR</th>
<th>$\sum_{UseR}^i$</th>
<th>MUSA $_SecR$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The system shall use symbols and words that are naturally understandable by the user community to prevent incorrect results</td>
<td>2</td>
<td>8.66667</td>
<td>2</td>
<td>9.66667</td>
<td>3.0555555556</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>8.66667</td>
<td>2</td>
<td>9.66667</td>
<td>3.0555555556</td>
</tr>
</tbody>
</table>

$$MUSA \_SecR_i = \frac{\left(\sum_{CSR}^i + \sum_{UseR}^i\right)}{3} + 2)$$

$$MUSA \_SEC = \sum_{i=1}^{MUSA} \ MUSA \_SecR_i$$

MUSA $\_SEC$ = 3.0555555556
B.2 Utility Functions
Figure B.5: Applying Utility Functions: This figure explains applying the utility functions in Chapter 3.
B.3 Decision Tree
Figure B.6: Applying the Decision Tree. This figure explains applying the decision tree in Chapter 3.
B.4 Assessment Results
Figure B.7: The Assessment Final Results. This figure shows the final results of assessing the requirements in Chapter 3.

<table>
<thead>
<tr>
<th></th>
<th>Security</th>
<th>Usable Security</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Before using AFUS</strong></td>
<td>0.426</td>
<td>No Assessment</td>
<td>0.574</td>
</tr>
<tr>
<td><strong>After using AFUS</strong></td>
<td>0.246</td>
<td>0.508</td>
<td>0.247</td>
</tr>
</tbody>
</table>
Appendix C

Institutional Review Board - IRB

This appendix shows the Institutional Review Board Form (IRB-Form) and its approval for conducting the comparative experiment of Section 6.4.
Notice of Exempt Review Status

From: Florida Tech Institutional Review Board
FWA00014339, Exp. 4/11/2017, IRB00001690

To: Yaseer Haquewi

Date: March 31, 2014

IRB Number 14-047

Study Title: Choice-based and traditional authentication: A usability comparative study

Dear Researcher:

Your research protocol was reviewed and approved by the IRB Chairperson. Per federal regulations, 45 CFR 46.101, your study has been determined to be minimal risk for human subjects and exempt from 45 CFR46 federal regulations and further IRB review or renewal unless you change the protocol or add the use of participant identifiers. This study is approved for one year from the above date. If data collection continues past this date, a Continuing Review Form must be submitted.

All data, which may include signed consent form documents, must be retained in a locked file cabinet for a minimum of three years (six if HIPAA applies) past the completion of this research. Any links to the identification of participants should be maintained on a password-protected computer if electronic information is used. Access to data is limited to authorized individuals listed as key study personnel.

The category for which exempt status has been determined for this protocol is as follows:

2. Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior so long as confidentiality is maintained.
   a. Information is recorded in such a manner that the subject cannot be identified, directly or through identifiers linked to the participant and/or
   b. Subject's responses, if known outside the research would not reasonably place the subject at risk of criminal or civil liability or be damaging to the subject's financial standing, employability, or reputation.
RESEARCH INVOLVING HUMAN SUBJECTS
Exempt Application

This form shall be used if there is minimal risk to human subjects and one or more of the conditions below apply. If there is more than minimal risk associated with the research (none of the conditions below apply) or if the research utilizes a special population (children, prisoners, institutionalized individuals, etc.), please use the full application form found on the IRB website.

You should consult the university’s document "Principles, Policy, and Applicability for Research Involving Human Subjects” prior to completion of this form. Copies may be obtained from the Office of Sponsored Programs and on the IRB website.

Name: Yasser Hausawi, William H. Allen, and Gisela Susanne Bahr
Date: 03/17/2014
Academic Unit: Computer Sciences
Email: yhausawi@my.fit.edu, wallen@fit.edu, gbahr@fit.edu
Title of Project: Choice-Based and Traditional Authentication: A Usability Comparative Study

☐ 1) Research conducted in established or commonly accepted educational settings, involving normal educational practices, such as:
   a. research on regular and special education instruction strategies, or
   b. research on the effectiveness of or the comparison among instruction techniques, curricula, or classroom management methods.

☐ 2) Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures, or observation of public behavior unless:
   a. the subjects can be identified, directly or through identifiers linked to the subjects and
   b. any disclosure of subjects’ responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects’ financial standing, employability, or reputation.

Note: This exemption does not apply to survey procedures or interviews involving minors.

☐ 3) Research involving the use of educational tests, survey or interview procedures, or observation of public behavior if:
   a. the subjects are elected or appointed public officials or candidates for public office or
   b. the confidentiality of the personally identifiable information will be maintained throughout the research and thereafter.

☐ 4) Research involving the collection or study of existing data, documents, records, or specimens if these sources are publicly available or if the information is recorded by the investigator in such a manner that subjects cannot be identified, indirectly or through identifiers linked to the subjects.

☐ 5) Research and demonstration projects that are conducted by or subject to the approval of Department or Agency heads and that are designed to study, evaluate, or otherwise examine:
   a. public benefit or service programs,
   b. procedures for obtaining benefits or services under those programs,
   c. possible changes in or alternatives to those programs or procedures, or
   d. possible changes in methods or levels of payment for benefits or services under those programs.

☐ 6) Taste and food quality evaluation and consumer acceptance studies if:
   a. wholesome foods without additives are consumed or
   b. food is consumed that contains food ingredients found to be safe by the Food and Drug Administration or approved by the Environmental Protection Agency or the Food Safety and Inspection Service of the U.S. Department of Agriculture.
1. List the objectives of the proposed project.

The study investigates effects of the proposed Choice-Based Authentication Approach (CBAA) on usable-security measures in human computer interaction. The approach is based on two concepts: allowing authentication according to end-user’s choice during the sign-up process for enhanced usability and universal access, and increasing penetration complexity for adversaries by showing all the authentication methods during the login process for better security. The experiment compares CBAA and the Traditional Single Authentication Approach (TSAA), which is generally associated with the use of alphanumeric passwords.

2. Describe the research project design/methodology. Discuss how you will conduct your study, and what measurement instruments you are using. If your project will use a questionnaire or structured interview, attach. Please describe your study in enough detail so the IRB can identify what you are doing and why.

The experiment is a between subjects design with 30 participants, who are randomly assigned to one of two condition. Each condition will use one of two authentication systems implemented for this study. The first system represents is the proposed CBAA that gives the user the choice of three different authentication methods: alphanumeric password based on the National Institute of Standards and Technology policy (NIST - SP 800 Series), graphical password based on the recall-based method, or use of biometrics (fingerprint). The second system represents the TSAA, which only uses the alphanumeric password based on the NIST - SP 800 Series. Subjects will be randomly assigned to authentication systems.

The dependent measures represent three usable-security properties (effectiveness, efficiency, and satisfaction) and are as follows: number of errors, task completion, signup time, login time, degree of positive affect, ease of use, and degree of negative affect.

In addition a brief survey will be administered at the end of the study to measure user satisfaction. We will use an already available survey called System Usability Scale (SUS). Figure 2 is a snapshot of the SUS survey.

The rundown of the experiment is as follows:

1. Welcome participant, introduce study, and administer consent form (5 minutes).
2. Randomly assign participant to one of two conditions (either the CBAA, or the TSAA) (2 minutes).
3. Conduct the interaction part of the session (practical) (8 minutes).
4. Break (5 minutes).
5. Administer the survey (5 minutes).
6. Debrief (5 minutes).
### System Usability Scale

**Instructions:** For each of the following statements, mark one box that best describes your reactions to the system today.

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th></th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I think that I would like to use this system frequently.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. I found this system unnecessarily complex.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. I thought this system was easy to use.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. I think that I would need assistance to be able to use this system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. I found the various functions in this system were well integrated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. I thought there was too much inconsistency in this system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. I would imagine that most people would learn to use this system very quickly.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. I found this system very cumbersome/awkward to use.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. I felt very confident using this system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. I needed to learn a lot of things before I could get going with this system.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Please provide any comments about this system:**

*Figure 1: A screenshot of the SUS questionnaire*

3. Describe the characteristics of the subject population, including number, age, sex, and recruitment strategy (attach actual recruitment email text, recruitment flyers etc).

The targeted subjects are undergraduate and graduate students from different disciplines. The reason for targeting students from different disciplines is to have enough subjects representing various security knowledge levels. There are no restrictions on subjects' age or gender as long as the age is over 18 years old. The targeted number of the subjects is 30 per each condition (the total number is 60 subjects).

We will adopt three recruitment strategies: electronic recruitment via electronic mail, physical recruitment by going to classes to introduce the study and recruit subjects, and using recruitment flyers through the university campus. Figure 3 shows the form used in all of the three recruitment strategies.
Usable-Security Authentication Study

Participants needed immediately

About the study: We have proposed an authentication approach and would like to investigate whether it helps achieving usable-security or not. This enjoyable study is open to all students of FIT.

Location: TBD.

Contact info: Yasser Hausawi  cell: 321-704-6610  email:yhausawi@my.fit.edu

Study Dates: TBD.

Sections Times: 9:00am / 9:45am / 10:30am / 11:15am / 12:00pm / 12:45pm / 1:30pm / 2:15pm / 3:00pm / 3:45pm / 4:30pm

How to sign-up:

Step 1: Please take contact information below and email Yasser about your most convenient days/times.

Step 2: Please arrive at least 5 minutes before your scheduled appointment.

Step 3: Please call or send an email if you need to reschedule.

Thank you

Figure 2: Universal Recruitment Sheet

4. Describe any potential risks to the subjects (physical, psychological, social, legal, etc.) and assess their likelihood and seriousness. Research involving children must carefully assess risks and describe the safeguards in place to minimize these risks.

The CBAA provides biometrics authentication (fingerprint) as one alternative choice. However, the system will neither extract fingerprint features nor store the data, as the fingerprint scanner is a dummy device. Moreover, the subjects will be completely anonymous after the session completion and there will not be any link to any subject. This information will be stated clearly to the participants at the beginning of each session in the consent form.
5. Describe the procedures you will use to maintain the confidentiality and privacy of your research subjects and project data.

The research team will provide a consent for participation form for the subjects to sign on and make sure that they have a copy of that form (see Figure 4). A specific time (that other subjects will not be available) will be allocated for only one particular subject in order to ensure subject’s privacy. The collected data will be stored in a digitally protected device (with username and password). The device will be stored in a physically protected location (secured Florida Tech. building). The collected data will be destroyed right after the statistical analysis is done.
6. Describe your plan for informed consent (attach proposed form).

The informed consent form (see Figure 4).

Consent for Participation in an Experimental Research Study

I volunteer to participate in an experimental research study conducted by Mr. Yasser Hausawi and his co-researchers from Florida Institute of Technology, Melbourne, Florida, USA. I understand that the study is designed to gather information about usable-security of an authentication approach. I will be one of approximately 60 people participating in this research study. The participation session will take approximately 25-30 minutes.

1. My participation in this study is voluntary. I understand that I will not be paid for my participation. I may withdraw and discontinue participation at any time without penalty, if I decline to participate or withdraw from the study, no one on my campus will be told.
2. I understand that most participants in this study will find the tasks interesting and thought-provoking. If, however, I feel uncomfortable in any way during the experiment session, I have the right to decline to perform any task and/or answer any question or to end the session.
3. Participation involves working on a custom-built system developed by the research team and filling a survey. The session will last approximately for 25-30 minutes. Notes will be taken by the investigator during the session.
4. A screen recording of the whole session will be taken. If I don’t want my session’s screen to be taped, I will not be able to participate in the study. RECORD SCREEN (initial)
5. I understand that the researcher will not identify me by name in any reports using information obtained from this participation, and that my confidentiality as a participant in this study will remain secure. Subsequent uses of records and data will be subject to standard data use policies, which protect the anonymity of individuals and institutions.
6. I understand that the fingerprint scanner is a dummy device, and will not store finger features.
7. Faculty and administrators from my campus will neither be present during my participation period nor have access to raw data. This precaution will prevent my individual text/audio comments from having any negative repercussions.
8. I understand that this research study has been reviewed and approved by the Institutional Review Board (IRB) for Studies Involving Human Subjects: Office of Research at Florida Institute of Technology. For research problems or questions regarding subjects, the Institutional Review Board may be contacted through Office of Research, Florida Institute of Technology, 150 W. University Blvd, Melbourne, Florida 32901, Phone (321) 674-0960, Fax (321) 674-0965.
9. I have read and understand the explanation provided to me. I have had all my questions answered to my satisfaction, and I voluntarily agree to participate in this study.
10. I have been given a copy of this consent form.

My Signature ___________________________ Date ___________________________

My Printed Name ___________________________ Signature of the Investigator

For further information, please contact:

Mr. Yasser Hausawi
Florida Institute of Technology
150 W. University Blvd
Melbourne, Florida 32901
Phone: 321-704-6610
Email: yhausawi@my.fiu.edu

Figure 3: Consent for Participation A Research Study
7. Discuss the importance of the knowledge that will result from your study and what benefits will accrue to your subjects (if any).

The goal of this work is enhance currently available authentication methods. Typical authentication systems provide one way for verifying users (passwords, ID cards, tokens, or biometrics). In some advanced authentication systems, dual authentication is used to verify users (passwords, and tokens or biometrics), but the user does not have the choice to select the most suitable method of interaction with the system. Both are required. Dual authentication increases the complexity of security systems to the level that makes penetration more difficult, but usability issues remain unanswered. For example: *Is such interaction complexity necessary to make authentication systems difficult to penetrate? Is there a simpler way to make security systems simultaneously usable and difficult to penetrate rather than adding complicated interaction designs that burden the user?*

To answer the above concerns, we have proposed the user-choice approach as a way to make security and usability synergetic in order to achieve an acceptable level of usable-security without increasing users’ cognitive load.

8. Explain how your proposed study meets criteria for exemption from Institutional Review Board review (as outlined on page 1 of this form).

The intended research has minimal risks to human subjects, who must be over 18 years old. Moreover, the intended research meets with the following conditions:

1. It involves the use of educational tests, such as cognitive load and task achievement, and survey procedures. However, the subjects can neither be identified directly nor through identifiers linked to the subjects. The subjects’ responses will not be disclosed outside the research and the subjects will not be at risk of any of the following: criminal, civil liability, financial standing damage, employability, or reputation.

2. The confidentiality of the personally identifiable information, if any, will be maintained throughout the research by protecting the research resources electronically and physically, and thereafter by destroying the data after the research is completed.
Signature Assurances

I understand Florida Institute of Technology's policy concerning research involving human subjects and I agree:

1. to accept responsibility for the scientific and ethical conduct of this research study,
2. to obtain prior approval from the Institutional Review Board before amending or altering the research protocol or implementing changes in the approved consent form,
3. to immediately report to the IRB any serious adverse reactions and/or unanticipated effects on subjects which may occur as a result of this study,
4. to complete, on request by the IRB, a Continuation Review Form if the study exceeds its estimated duration.

PI Signature: ___________________________ Date: 03/17/2014

Advisor Assurance: If primary investigator is a student
This is to certify that I have reviewed this research protocol and that I attest to the scientific merit of the study, the necessity for the use of human subjects in the study to the student's academic program, and the competency of the student to conduct the project.

Major Advisor: ___________________________ Date: 03/17/2014
Major Advisor (print): William H. Allen Ph.D

Academic Unit Head: It is the PI's responsibility to obtain this signature
This is to certify that I have reviewed this research protocol and that I attest to the scientific merit of this study and the competency of the investigator(s) to conduct the study.

Academic Unit Head: ___________________________ Date: 03/17/2014

FOR IRB USE ONLY

IRB Approval: ___________________________ Date: 3-31-14

Name: ___________________________ 19-047

IRB #

Florida Tech IRB: November 2005
Appendix D

Determining the Optimal Balance between Security and Usability

This appendix presents the depiction of using the trade-off chart of queuing theory as a future direction in order to determine the optimal balance of the trade-off between security and usability (described in Section 7.3).
Figure D.1: All Possible Values of Usable-Security. This figure shows all the possible cases at which security and usability can interact based on Figures 5.2 and 5.3 in Chapter 5.
Figure D.2: The Possible Trends of Security and Usability Values. This chart displays the Trend-values of both security and usability.
Figure D.3: The Trade-Off Chart. This chart shows the possible optimal balance between security and usability after calculating the exponential of each trend. The optimal balance between security and usability is determined based on the calculation of usable-security. This shows all the possible cases at which security and usability can interact based on Figures 5.2 and 5.3 in Chapter 5.