Enhancing Cybersecurity by Generating User-Specific Security Policy through the
Formal Modeling of User Behavior

by
Arwa AlQadheeb

Bachelor of Science
in Computer
University of Dammam
2014

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

Masters of Science
in
Computer Science

Melbourne, Florida
January, 2020
The author grants permission to make single copies.
We the undersigned committee hereby approve the attached dissertation

Enhancing Cybersecurity by Generating User-Specific Security Policy through the Formal Modeling of User Behavior by Arwa AlQadheeb

______________________________
Siddhartha Bhattacharyya, Ph.D.
Assistant Professor
Computer Engineering and Sciences
Committee Chair

______________________________
William H. Allen III, Ph.D.
Associate Professor
Computer Engineering and Sciences
Committee Member

______________________________
Meredith Carroll, Ph.D.
Associate Professor
Aeronautics
Outside Committee Member

______________________________
Samuel J. Perl, M.S.
Software Engineering Institute at Carnegie Mellon University

______________________________
Philip J. Bernhard, Ph.D.
Associate Professor and Department Head
Computer Engineering and Sciences
ABSTRACT

Title:
Enhancing Cybersecurity by Generating User-Specific Security Policy through the Formal Modeling of User Behavior

Author:
Arwa AlQadheeb

Major Advisor:
Siddhartha Bhattacharyya, Ph.D.

Despite the ongoing efforts to develop cutting-edge security solutions, the question always remains whether these technologies can overcome system vulnerabilities that often result from poor security practices made by end-users. Recently, some research devoted to study the human role in cybersecurity, especially the psychological aspect. Researchers found that the users’ responses to security-related situations correlate with various elusive factors such as demographics, personality traits, decision-making styles, and risk-taking preferences. That explains why some users neglect to act according to common security tips and advice. The goal of this research is to make cybersecurity maintain a high-level of quality and reliability; we reinforce the policy generated to overcome weaknesses created by the human link in the security chain. To achieve this goal, we developed a formal method-based approach to model and examine end-users security-related behaviors described by Finite-State Automaton (FSA). The methodology initially assesses the cybersecurity behaviors that users exhibit during the use of electronic devices such as laptops, smartphones, or the access to Internet accounts in daily life, spanning four aspects: device securement, password generation, proactive awareness, and updating. Once we identified these behaviors, we created a knowledge repository to represent the behavior using Finite-State Automata (FSA). This enabled
the formulation of linear-time security properties based on Timed Computation Tree Logic (TCTL) to check the reachability of good and poor behaviors. To perform the reachability analysis, we used a model checking tool to identify poor security behaviors. This supports our objective to distinguish which user lacks security knowledge and awareness and regarding what aspect, and generating effective countermeasures. Hence, we were able to determine what type of policies that an organization or other entity should impose on specific users based on their security decisions. Our approach assumes a Zero Trust philosophy but may help in other security systems to mitigate the risks of being exposed due to ignorant cyber users or targeted by cyber offenders.
# Table of Contents

Abstract iii  
List of Figures vii  
List of Tables viii  
Abbreviations ix  
Acknowledgments x  
Dedication xii  
1 Introduction 1  
2 Related Work 6  
3 Research Methodology 15  
4 Selecting Predictor 17  
5 Modeling User Security Behavior 21  
  5.1 Model Considerations 22  
  5.1.1 Levels of Abstraction 23  
  5.2 Model Paradigm 24
### 6 Generating Linear-Time Security Properties

- **6.1 Overview of Linear-Time Properties**
  - 6.1.1 Invariant
  - 6.1.2 Liveness Properties
- **6.2 Specification Language in Uppaal**
- **6.3 Knowledge Base Properties**
  - 6.3.1 Device Securement
  - 6.3.2 Password Generation
  - 6.3.3 Proactive Awareness
  - 6.3.4 Updating

### 7 Generating User-Specific Security Policy

- **7.1 Automated Analysis**
  - 7.1.1 Representing Test Cases
  - 7.1.2 Generating User-Specific Security Properties
  - 7.1.3 Reachability Analysis
- **7.2 Generating User-Specific Policy**

### 8 Conclusion

- Appendix
# List of Figures

3.1 User-specific policy generation framework ........................................ 16

5.1 Knowledge base architecture ....................................................... 22

6.1 AG computation tree ............................................................... 33
6.2 EG computation tree ............................................................... 33
6.3 AF computation tree ............................................................... 34
6.4 EF computation tree ............................................................... 34
6.5 Device securement state-transition graphs ........................................ 37
6.6 Password generation state-transition graphs ...................................... 41
6.7 Proactive awareness state-transition graphs ...................................... 44
6.8 Updating state-transition graphs .................................................. 48

7.1 Tom’s state-transition graph ....................................................... 51
7.2 Schematic view of the model-checking approach ................................ 54
7.3 Example (1) of Tom’s reachability analysis results ............................ 56
7.4 Example (2) of Tom’s reachability analysis results ............................ 56
List of Tables

4.1 Comparison between [22] and [27] psycho-metrics . . . . . . . . . . . 18
4.2 Correlations between SeBIS and GDMS dimensions from [22] . . . . 19
4.3 Correlations between SeBIS and GDMS dimensions from [27] . . . . 20

6.1 Temporal connectives in Uppaal . . . . . . . . . . . . . . . . . . . . . 33
6.2 Device securement properties . . . . . . . . . . . . . . . . . . . . . . . 37
6.3 Password generation properties . . . . . . . . . . . . . . . . . . . . . . 40
6.4 Proactive awareness properties . . . . . . . . . . . . . . . . . . . . . . 43
6.5 Updating properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

7.1 Tom’s reachability analysis results . . . . . . . . . . . . . . . . . . . 55

8.1 John’s reachability analysis results . . . . . . . . . . . . . . . . . . . . 74
8.2 Sara’s reachability analysis results . . . . . . . . . . . . . . . . . . . . . 75
8.3 Bella’s reachability analysis results . . . . . . . . . . . . . . . . . . . . . 76
8.4 Zac’s reachability analysis results . . . . . . . . . . . . . . . . . . . . . . 77
8.5 Zoe’s reachability analysis results . . . . . . . . . . . . . . . . . . . . . . 77
# List of Abbreviations

| Abbreviation | Full Form |
|--------------|-----------|
| FSA          | Finite-State Automata |
| SeBIS        | Security Behavior Intentions Scale |
| USA          | United State of America |
| ISP          | Internet Service Providers |
| US-CERT      | United States Computer Emergency Readiness Team |
| MTurk        | Amazon Mechanical Turk |
| CFC          | Consideration for Future Consequences |
| DoSpeRT      | Domain Specific Risk Attitude |
| IPIP         | International Personality Item Pool |
| GDMS         | General Decision Making Style |
| BIS          | Barratt Impulsiveness Scale |
| NFC          | Need For Cognition |
| USPGF        | User-Specific Policy Generation Framework |
| TA           | Timed Automaton |
| TCTL         | Timed Computation Tree Logic |
| CTL          | Computation Tree Logic |
| DBIR         | Data Breach Investigations Report |
| BFS          | Breadth-First Search |
| DFS          | Depth-First Search |
| DS           | Device Securement |
| PG           | Password Generation |
| PA           | Proactive Awareness |
| U            | Updating |
Acknowledgements

First and foremost, all praise and thanks are due to almighty Allah, the Most Merciful, the Most Compassionate, and the Most Lovingly Beneficent. I must acknowledge that this work would never have seen the light of day without the help and guidance of the Lord of Majesty and Generosity, Allah.

Second, I owe an enormous debt of gratitude to my beloved country, the Kingdom of Saudi Arabia, for providing me the opportunity to pursue my higher education and achieve one of my biggest dreams in life.

Third, I am deeply indebted to my thesis advisor Dr. Siddhartha Bhattacharyya for his vital support, significant intellectual guidance, and infinite patience. He was a great mentor who helped me go through all the difficulties and challenges I came across while I was working on my Master’s study, and for that, I do not have enough words to express my sincere appreciation and gratitude.

Additionally, I highly value the efforts expended by Mr. Samuel J. Perl, who generously offered invaluable advice and suggestions; I truly appreciate his willingness to help and participation in every possible way.

Further, a heartfelt thanks go to Dr. William H. Allen III, my academic advisor, for keeping his office door always open to me whenever I needed advice on my Master’s program or any other matter. Also, an exceptional thanks to Dr. Meredith Carroll
for taking time out of her busy schedule to serve as a member on my thesis advisory committee.

Last, but not least, many thanks to my supportive friends for being around during the ups and downs of this beautiful journey, and to every other individual who helped in one way or another to make this course of my life easier and more enjoyable.
Dedication

This work is dedicated to my great parents, Jassim AlQadheeb and Maisa AlWheeb, for their unconditional love, generous support, constant prayers, and motivational encouragement. If it were not for Allah then them, I would not make it this far, and I would not be the person I am today. They are the source of my inspiration, strength, and life, and for that, I am very grateful.

Additionally, I would like to express my deepest gratitude and appreciation to my sisters Maha and Mai. I cannot even begin to describe how thankful I am to have them in my life; they were there for me every step in the way from day one. They continuously pushed me to be the best version of myself, and they always were the shoulder I needed to lean on during tough times. Plus, to my beloved brothers Mohammed, Musab, and Muath, my niece Maria, and to my one and love my cat Lowy.

I love you all dearly.
Chapter 1

Introduction

The level of innovation in today’s computing landscape changed rather markedly in the past few years. The vast use of different emerging technologies such as Internet of Things (IoT), cloud computing, and artificial intelligence, by organizations and communities, has threatened security and assurance within information systems. Cyber defenders find themselves in an ongoing war with cybercriminals and threat actors who continue to find ways to launch their attacks more aggressively. Attackers practicing manipulative tactics and using available sophisticated technologies made old-fashioned means of protection relatively ineffective to combat increasingly advanced and crafty attacks. Security professionals remain determined to develop more robust and reliable security solutions in order to be more prepared to deal with cyberattacks. In 2010, John Kindervag, a Security and Risk Principal Analyst at Forrester Research Inc., developed a Zero Trust security model that has a different take on security and revolves around classic security principles, the most important of which is “never trust, always verify.” [34]. The assumptions about how a Zero Trust network should behave or operate are different from traditional security models in terms of rules of operation that reflect improvements
in adversary strategies, techniques, and observed behaviors. In a Zero Trust environment, organizations have no default trust in any user or any device, whether inside or outside the network’s perimeter, ensures that every party is subject to restricted access with continuous authentication and authorization, and verifies that every action made by that party is permitted. For instance, if an intruder was able to get sustained access to an organization’s network to reach valuable assets and confidential data, then any lateral movement through the network is more likely to be detected and denied due to the internal strict security access control policy [34].

To better minimize the risk of being exposed to cyberattacks, organizations considering Zero Trust should also pay close attention to the human aspect that is often the most vulnerable component of any security system [27]. The natural structure of humans psychology, the limitation of humans’ information processing capacity, and their almost absolute reliance on previous experiences stand in the way of making the right choices [61]. Even though the cost of a breach that results from a human error or a system malfunction is occasionally less expensive than the cost of a breach caused by a cybercriminal or a malicious insider, this should not diminish the consequences of human negligence [53]. Between the period of July 2018 and April 2019, the Ponemon Institute conducted a study based on interviews with 507 companies across 16 geographies that suffered data breaches. The study reveals that the negligence of users is the leading cause of 24% of data breaches, which are worth, on average, around $3.5 million of financial damage. Unintentional and accidental breaches result from so-called inadvertent insiders who had experienced a successful phishing attack or had their devices infected, lost, or stolen. According to the Ponemon Institute, human errors in cybersecurity cost $133 on average per compromised record, and it takes organizations around 181 days and 61 days identify and contain, respectively, a data breach that is related to
such careless human action [53].

Most computer systems are designed based on the general perception of the average user as if all users are all the same, overlooking the fact that individuals markedly differ from each other [22][31]. Because there are individual differences that separate oneself from another and contribute to shaping each individual’s form of self [9], one can consequently justify why some people respond and interact differently to one computer system. Recently, in the context of human factors in cybersecurity, researchers and practitioners have begun to examine the psychology of humans to understand users’ behaviors toward privacy and security, bearing in mind their differences. Some works showed that individual differences in demographics and psychological constructs (e.g., personality traits, decision-making styles, and risk-taking preferences) have a significant relationship with security behaviors and privacy attitudes. For instance, the geographical region of individuals influences their Internet browsing habits; Norwegians and Japanese tend to browse the Internet more cautiously than their counterparts in Italy and Spain [14], which in turn lower their chances of being exposed to malware-infected websites. Also, differences in personality and gender correlate with users’ susceptibility to cyberattacks. Females with a specific type of personality are more likely to be victimized by a cyberattack, such as social-engineering or email-based phishing malicious activities, more than males or other females with different personalities [3][17][30].

Groups responsible for information systems should help and guide users to make better security decisions. Although considerable research devoted to investigate the correlation between the various traits of personality and users’ security-related behaviors, other variables of individual differences gained somewhat less attention, namely decision-making. Analyzing the decision-making process of humans in cybersecurity is an ideal strategy that explains how to improve the abnormalities in users’ behaviors.
The fact is that no matter how robust and sophisticated a security system is, computer users could pose a threat to an organization’s data either by complacency, ignorance, or misplaced trust. In 2016, a spear-phishing scam compromised the personal information of several employees at Snap Inc. A scammer claiming to be Evan Spiegel, the Chief Executive Officer of Snapchat, was able to deceive an employee at the payroll division asking for coworkers’ payroll information. Unfortunately, the targeted employee was not vigilant enough to question whether it is a fraudulent scheme or not before sending the requested information to the impersonator [3]. Cybercriminals deliberately deceive and manipulate users to make security mistakes by exploiting their cognitive weaknesses and lulling them into a false sense of security. In most cases, attackers have superior technical knowledge that they attempt to trick computer users with.

The prominent convicted hacker Kevin D. Mitnick, known for his high-profile arrest in 1995 for various computer and communication crimes and now runs the security firm Mitnick Security Consulting, LLC., states that the most effective technique to break into a company’s system is to try exploiting the weakest link that is humans. From his point of view, the only reliable approach to overcome this problem is to combine security technologies with firm security policies along with proper education programs and training sessions for users [42]. As long as users are the weakest link in cybersecurity, cunning adversaries will continue to seek out and exploit users’ vulnerabilities in almost any information security system via every devious possible way. The issue with Mitnick’s solution is how to automate the process of having security policies that help with users’ security misbehavior and poor decisions. In our research, we offer an ideal approach to impose more control over the user by analyzing the security behavior and then generating user-specific policy within a Zero Trust environment. Our goal is to make cybersecurity maintain a high-level of quality and reliability; we reinforce
the policy generated to overcome weaknesses created by the human link in the security chain. The question we address in our research is: How to automatically and satisfactorily generate fair and explicit security policy after observing and analyzing security behaviors, especially security-related decisions, exhibited by end-users in an environment with Zero Trust assumptions? This question breaks down into three component questions:

**Q1:** Is it possible to model end-user behavior using Finite-State Automaton (FSA)?

**Q2:** Is it possible to generate linear-time security properties and then verify the reachability of good and poor behaviors?

**Q3:** Is it possible to compute the type of security policy that should be imposed on a specific user using the developed formal method that results from 1 and 2?

The answer to the questions mentioned above present a preliminary automated method for generating a user-specific security policy that suggests a new contribution to the extant literature. We further enhance cybersecurity by distinguishing ignorant or negligent users who exhibit poor security practices and pose the most significant security risk on the information system.

The remainder of this research organized as follows: Chapter 2 describes in greater depth some related works and background information. Chapter 3 presents the research methodology, outlining the sequential development process for the automated approach. Chapter 4 identifies the requirements to model user security behavior. Chapter 5 discusses the model considerations and paradigm. Chapter 6 demonstrates the process of generating linear-time security properties. Chapter 7 defines the rule of generating a user-specific policy. Finally, Chapter 8 states our conclusions and future work.
Chapter 2

Related Work

The study of human factors psychology can lead to a deeper and broader understanding of human-computer interaction, which in turn contributes to make the security system more suitable as well as more convenient for users. This area of knowledge examines how the psychological factors of humans relate to their concerns about the privacy of information, and their compliance with the standard security policies and procedures.

In [60], West (2008) emphasizes the benefits of understanding the process of how people wrap their minds around security matters and make decisions. He says that system design and appearance do not change a thing about security as long as end-users persist in dismissing security warning dialogs, using a weak set of security options, and overstepping security policies and standards by accident. The article claims that humans are driven by fundamental ideas that control the way they visualize security in everyday life and the way they unintentionally compromise security. Individuals incline to think that they are more cautious, which implies the belief that they are less prone to risk than others. Computer users usually think that they have a million-to-one chance to suffer damages because of cyber products. They believe that their engagement in hazardous
activities would not matter because they already took intensive security precautions, according to the risk homeostasis theory, which suggests that people adapt their behavior to maintain an equivalent risk tolerance level [60].

Faulty decisions and practices concerning security also arise from the fact that users can make rash decisions by relying on similar past events with assumptions that no longer work [60]. According to the author, among other reasons, safety is just an abstraction, thereby less appealing to users during the analysis of cost-benefit trade-off such that the costs that result from impulsive security decisions are concrete, instant, and time-consuming. The benefits, on the other hand, are abstract, hard to measure, and much more long-termed. Some of the strategies that the author mentions for improving security-related behavior include rewarding pro-security decisions, increasing users’ awareness about risk, and committing to catching those who violate security policies [60].

According to West, Mayhorn, Hardee, and Mendel (2009), there are human and cognitive factors that influence users’ interactions with the security system and decisions towards security matters. The writers state that humans are exploitable, and there is much evidence that proves this fact. They mention that a report released by the IT Policy Compliance Group reveals that the majority of companies, which suffered incidents of data abuse or theft, acknowledge that internal workers were the cause of the problems. Few of them attribute the reason for malicious activities such as hacks, attacks, and malware. West et al. (2009) express that end-users are behind almost all security-related incidents because of their slow technological adoption, their misconception about risk and its impact on their behaviors, and their poor decision-making tactics regarding security.

The human decision-making process is not at its best in situations that involve reflection on risks because there are factors that affect the quality of decisions. Those
factors combine internal agents such as foreknowledge or previous experience and environmental agents such as lack of time or situational status [61]. The authors add that comprehending the principles that influence decision-makers helps to create interferences, which would improve the result of their decisions. They declare that there are psychological aspects that justify the reason why people have some form of an impaired decision-making process. One of these aspects is that the capacity of information processing in humans makes them decide quickly based on what seems satisfactory enough. The other aspect is that humans make decisions out of the memory of past experiences and often fail to evaluate all currently available options [61].

Herley (2009) believes that the claim, which states that Internet users are careless and unmotivated when it comes to security advice, is mistaken and overstated. Indeed people are known for their bad reputation in taking no account of security advisories, preferring easy-to-remember passwords, and most importantly, they are at the top level of inattention about the errors of certificates. The author argues that the reason users are unwilling to behave according to security “advice” is justifiable from an economic perspective. Although security guidance offers users some level of protection from the costs of cybersecurity attacks, it also brings a considerable increase in effort. He thinks that security advice and recommendations available to users are often complicated and abstruse, and their benefits are usually conjectural or questionable. For instance, nearly all passwords’ advice is obsolete and unclear regarding the consequences of choosing weak passwords. Plus, almost all, if not all, of certificate errors are just false alarms [32].

Herley (2009) indicates that the reason that users continue to reject matters of security is because time cost that they would have to incur if they took the time inspecting URLs to avoid phishing. Indeed security advice protects against cybersecurity issues,
but it comes with a heavy burden of time and effort. Generally, users engage in carrying out a cost-benefit analysis before they apply security advice. They look at the cost of the effort they will put in to follow the advice while the benefit is avoiding the harm that could likely arise from an attack. The harm that results from a cyber attack could be monetary cost, or also effort and time spent to resolve the issue [32]. Harley (2009) explains that although the article is argumentative, and justifying on behalf of users the reason for ignoring security advice, one should not wind up thinking that all security advice is troublesome. On the contrary, it is helpful, but its only flaw is that it does not properly balance out costs and benefits.

Any aware user would realize that the cost of following security advice is far higher than its benefit. In order to change this outcome, there has to be a more compelling trade-off. Harley (2009) suggests that one of the solutions is that professionals and practitioners, who offer security advice, must have a better comprehension of harms that users genuinely endure. He also adds that security experts should dismiss and overlook the least favorable advice because security advice that was in use long ago cannot address the problems users are currently encountering. The other way of change is through dignifying users’ effort and time and considering them. It is by noting that overemphasizing the harm of one attack reduces the time and effort that the user pays on other attacks, meaning that when security professionals exaggerate all attacks, users resolve to pay attention to none [32].

Egelman and Peer (2015) express that corporations spend significant resources on preparing workers to interact with information security systems appropriately before they are allowed to access the internal network of the corporation. The authors state that the evaluation and assessment techniques are integral and necessary parts of every security training or education program. Just about all organizations offer end-users the
fundamental concepts of cybersecurity through training and education sessions as well as overloading them with advice and tips on how to ensure security while accessing and using the system. By doing so, end-users wind up confounded and disoriented and become trapped in a “bowl of mixed information,” lost between which advice to follow and which to ignore. It is for this reason the authors recommend analyzing the general behavior of users as an attempt to determine specific advice that is suitable for each user, which would make it easier to follow and remember [23].

Egelman and Peer (2015) find that the use of a scale is one of the most traditional valuable tools for studying human behaviors’ instead of carrying out observational or behavioral experiments, which could be too exorbitant in cost for organizations, especially those with limited budgets. Scales can effectively and efficiently estimate the changes that occurred in end-users’ behaviors and attitudes over some time. Egelman and Peer developed a scale named the Security Behaviour Intentions Scale (SeBIS) that evaluates the computer users’ ability to adhere to computer security advice based on self-declared information.

The researchers’ scale focuses on four constructs which are: device securement, password generation, proactive awareness, and updating. Device securement includes setting up lock screen using patterns, passwords, or PINs. Password generation relates to the practice of creating secure passwords. Proactive awareness is about spotting suspicious signs of phishing scams. Lastly, updating means keeping software up-to-date. Egelman and Peer specified those concepts of security after assessing the most common security-related advice suggested to Internet users by nationwide USA Internet Service Providers (ISPs), United States Computer Emergency Readiness Team (US-CERT), and industry associations. The study shows that further research can utilize the SeBIS scale to define users’ engagement with network security matters adequately, and therefore,
could aid end-users to better respond to various cybersecurity measures [23].

In [22], Egelman and Peer (2015) say that today’s systems do not take into account the differences in users’ characteristics. Developers build a design system to meet the majority of the user population. It is about time that system developers began to make the best use of their designs and consider individual differences. They mention that in the field of psychology, researchers have indicated that individual differences have some effect on people’s decision-making process. The article suggests that since individual differences constructs can show individuals’ perspective towards risk, then they can also foretell their privacy preferences. Experimenters could then develop the scales for these implicit factors to assist in determining the privacy preferences of individuals without the cost of asking them in person. The article also proposes that system design specialists should use measures of other constructs that influence decision-making to customize security warning systems. The writers justify that the reason for studying differences between decision-makers is to have a better knowledge of the effects that are more noticeable in a person who scores low or high on particular individual attributes scales. For instance, the article suggests that people who have weak skills with numbers and mathematics are the most affected by mood and presentation of information [22].

The experimenters agree with the claim that differences between people’s attitudes are a reasonable explanation of why their perceptions towards risks also vary. Systems should be created and developed with consideration to users’ different characteristics that further should be part of customizing privacy and security mitigations. Egelman and Peer (2015) dispute the Big Five personality model as a useful guide for grasping the privacy preferences of different people. They explain that in addition to using the five dimensions of the Big Five model, evaluating an individuals’ decision-making styles, and risk-taking preferences can inform about their privacy attitudes too. They
consider that the Security Behavior Intentions Scale (SeBIS), mentioned earlier [23], is an efficient measure to examine security behavior intentions towards device security, password generation, proactive awareness, and updating. Evaluating people’s differences concerning risk-taking preferences and decision-making styles shows strong correlations with their privacy attitudes, which is stronger than the correlations with Big Five personality model [22].

Halevi et al. (2016) point out that online threats have the entire information security industry on high alert. Some of the weaknesses that currently exists in the computer environment stems from the fact that system developers shape their designs based on general users in mind instead of designing them to match a specific user. The authors propose that the initial step towards coming down with security safeguards tailored to a specific user requires developers first to recognize the aspects that impact secure online behavior. To do so, they conducted a study to investigate the relationship that exists between variables of culture, personality trait, and users’ demographics and cybersecurity behavior. They found out that culture is a strong factor that influences how people preserve their privacy in cyberspace. Among the study findings, participants from the United States of America (USA) reveal much more information about themselves online compared to their counterparts from other countries [7]. More evidence that shows culture can impact privacy attitude is the fact that in USA, online banking is preferable to people than in other countries. Therefore, the authors suggest that creators should consider culture as a factor to predict user’s privacy attitudes in the system development process.

In [7], the researchers also observed how personality trait affects security attitude. They mention that individuals’ personality relates to their responses to various kinds of incentives that exist in their surroundings and their reactions to various situations.
The authors utilize the Big Five framework to explore how different personality traits influence security perceptions and intentions. The authors’ study reveals that conscientiousness strongly correlates with security behavior. It means that end-users who are thorough and meticulous in their work are more vigilant about security in their online engagements. Openness was also another predictor of users who are confident and capable of combatting cyberattacks. Neuroticism had a negative correlation with self-sufficiency. The more individuals show neuroticism, the less likely they are going to engage in secure online behavior. For demographics, Halevi et al. (2016) indicate that gender significantly and inversely relates to self-sufficiency and privacy attitudes, respectively. In which males have more self-confidence about limiting cybersecurity risks in comparison to females, which implies that men are more concerned about their privacy in cyberspace. The study further highlights risk perception as a determinant of both safe online behavior and self-sufficiency. People who previously encountered cyberattacks are more inclined to practice security properly and have more certainty about their security mitigation ability [7].

Gratian, Bandi, Cukier, Dykstra, and Ginther (2017) discuss the fact that the different characteristics of individuals influence their online security behavior. According to the article, the human element in cyberspace is the weakest link and the root cause of many cybersecurity attacks. The need for investigating the correlation between individual differences and security behavior intentions has become far more interesting in the cybersecurity domain. Fulfilling that need would help security practitioners with identifying individuals who are more likely to engage in risky security practices than others. The writers demonstrate that this knowledge is useful for improving the quality of current educational programs in order to enlighten those who put up the most significant cyber challenge. The authors conducted a study to explore the relationship between
cybersecurity intentions and the four main individual categories that constitute individual differences. These include personality traits, demographics, decision-making styles, and risk-taking preferences. The work of the study is built on previous research by Egelman and Peer [22] who were among the first who looked into individual differences as predictors of security and privacy attitudes [27].

Bandi et al. (2017) substantiate the accuracy of SeBIS and broaden SeBIS to examine the correlations of personality traits and demographics with security intentions. They conclude that the influence of individual differences on security behavior is environment-specific. The demographics that are influencing factors on cybersecurity are gender, age, citizenship, and length of employment. The study reveals that individuals of different professions have different perceptions when it comes to cybersecurity. Extroverted individuals are keener about security. Concerning gender, the experimenters found out that women tend to choose weaker passwords than men, as well as individuals between the age of 18 and 25. Women are also easy to fall victims to phishing scams more than men [27], corresponding to the same finding of Egelman and Peer [22].

The research states that proactive awareness of security varies according to the various set of demographics. It shows that females have less proactive awareness. Proactive awareness varies with various ages. Those between the age of 18 and 25 exhibit the weakest proactive awareness. Regarding keeping software up-to-date, males are more careful and vigilant than females. Among the 9% of respondents, females exhibit the weakest security practices compared to all other demographic groups included in this study [27].
Chapter 3

Research Methodology

This chapter describes the approach of developing and verifying a formal model of users’ security behaviors, which we later use to determine the type of policy that should be automatically generated and then imposed on that specific user. There are plenty of options available for a formal model development approach, differing in complexity from simple to incredibly complex; this can make choosing the best approach a daunting task. The development approach intends to address the issue of some poor security practices that are prevalent among Internet users and further maintain a high-level security system. To have that one approach that fits our goal and have control over how the development process proceeds, we decided to create our own approach to ensure that we precisely implement every detail within the process. Before going ahead with applying our approach and developing the intended formal model, we built an approach framework that helps us remain focused and organized while achieving our goal.

Our comprehensive User-Specific Policy Generation Framework (USPGF) in Figure 3.1 sets out the structure of this research as a whole, the figure shows the development process as a chain of five stages. We have broken some stages into other parts, and
each part includes more detailed information about the tools and the techniques we used to develop and verify the formal model. We divided our USPGF to identify and define the various concepts we examine and the relationship between those concepts. For each stage of the USPGF, Chapter 4, 5, 6, and 7 explain what is needed to improve a secure cyber environment through automatically analyzing given user behavior in order to distinguish threatening security practices, and then conclude the appropriate security policy that fits that specific user.

![Diagram](image)

Figure 3.1: User-specific policy generation framework

The User-Specific Policy Generation Framework (USPGF) identifies specific user behavior and uses a formal method-based approach to model the behavior and to generate security policies. Having an utterly logical framework approach establishes a clear vision for the direction of the research that puts first things first, which are selecting the most relevant candidate predictor of individual differences and examining its correlations with users’ security behaviors. Through the initial stage of USPGF, we can specify what behaviors or practices we must analyze and how they relate to endangering cybersecurity.
Chapter 4

Selecting Predictor

In this chapter, we formalize user’s security-related behavior by taking into account security decisions made by that user. It describes the scope of data collection, comparison, and selection in order to construct a reliable knowledge base that consists of the requirements set for developing the formal model, which we did after reviewing a wide range of possibilities.

As shown in Chapter 2, previous studies have comprehensively investigated the correlations between individual differences in demographics, personality traits, decision-making styles, and risk-taking preferences and their influence on users’ security-related behaviors in cyberspace. We chose to focus on decision-making because it is experimentally prove to be a stronger predictor of security practices than demographics, personality traits, and risk-taking.

The motivation of this research derived from the observation of two particular earlier studies that are [22] and [27], which were conducted by Egelman and Peer and Gratian, Bandi, Cukier, Dykstra, and Ginther, respectively. Both pieces of research aim to develop tailored security defenses, taking into account differences between individuals,
in order to limit users’ errors and their resulting consequences. The sample population of [27] included higher education participants from a large public university located in the USA, while Egelman and Peer surveyed a group of individuals from Amazon Mechanical Turk (MTurk) who are over the age of 18. They both made use of SeBIS to carry out their studies; however, Egelman and Peer’s area of interest regarding security was investigating the correlations between SeBIS and decision-making psycho-metrics. Gratian et al. did an extended version of [22] such that they substantiate the accuracy of SeBIS, broaden the use of SeBIS to examine the correlations of personality traits and demographics with security intentions. Researchers used different sets of metrics to conduct their experiment, illustrated in Table 4.1.

Table 4.1: Comparison between [22] and [27] psycho-metrics

| Metric                                      | Study [22] | Study [27] |
|---------------------------------------------|------------|------------|
| Consideration for Future Consequence (CFC) | ✓          | ✗          |
| Security Behavior Intentions Scale (SeBIS)  | ✓          | ✓          |
| Domain Specific Risk Attitude (DoSpeRT)    | ✓          | ✓          |
| International Personality Item Pool (IPIP)  | ✗          | ✓          |
| General Decision Making Style (GDMS)       | ✓          | ✓          |
| Barratt Impulsiveness Scale (BIS)          | ✓          | ✗          |
| Need For Cognition (NFC)                   | ✓          | ✗          |

We can see that both studies use SeBIS, DoSpeRT, and GDMS to serve the same purpose of identifying the relation between security intentions, risk-taking, and decision-making. The process of determining how to respond to a particular situation requires risk evaluation, considering future consequences, and exploring possible alternatives. The natural structure of humans psychology, the limitation of humans’ information processing capacity, and their almost absolute reliance on previous experiences stand in the way of making the right choices [61]. Failing to determine what is the right thing to do
and vice versa is a critical matter in cybersecurity. If some individuals decided to leave
their device unattended in public, set up a password that is easy to guess or hack, rush to
download an anonymous email attachment, or underestimate the importance of software
updates, they are more at risk to be victimized by a crafty individual.

We consider modeling security behaviors with each user’s behavior a sequence of
different connected decisions. Each decision correlates with particular SeBIS construct:
device securement, password generation, proactive awareness, or updating. Using the
findings that we collected from [22] and [27], Tables 4.2 and 4.3 shows the correlations
between SeBIS and GDMS, which measures decision-making across five dimensions:

- Rational: Evaluating situations logically before making decisions.
- Dependent: Counting on and looking at others to make decisions.
- Avoidant: Delaying and postponing making decisions.
- Intuitive: relying on instincts to make decisions.
- Spontaneous: Making rapid and hasty decisions.

Table 4.2: Correlations between SeBIS and GDMS dimensions from [22]

|                     | Rational | Dependent | Avoidant | Intuitive | Spontaneous |
|---------------------|----------|-----------|----------|-----------|-------------|
| Device Securement   | ✗        | ✗         | −0.133** | ✗         | ✗           |
| Password Generation | 0.145**  | ✗         | −0.220** | ✗         | ✗           |
| Proactive Awareness | 0.224**  | −0.157**  | −0.230** | ✗         | ✗           |
| Updating            | 0.229**  | ✗         | −0.247** | ✗         | −0.129*     |

*ρ < 0.005, **ρ < 0.001
Table 4.3: Correlations between SeBIS and GDMS dimensions from [27]

|                    | Rational | Dependent | Avoidant | Intuitive | Spontaneous |
|--------------------|----------|-----------|----------|-----------|-------------|
| Device Securement  | 0.164**  | ❌         | ❌        | ❌         | ❌          |
| Password Generation| ❌        | ❌         | -0.149*  | ❌         | ❌          |
| Proactive Awareness| 0.135    | -0.108    | -0.157** | ❌         | ❌          |
| Updating           | 0.153**  | ❌         | ❌        | ❌         | 0.243***    |

*ρ < 0.005, **ρ < 0.01, ***ρ < 0.001

Although both studies used the same independent variables (i.e., rational, dependent, avoidant, intuitive, and spontaneous) and the same dependent variables regarding SeBIS, it is very evident that the findings of [22] and [27] are heterogeneous, where predictors differently impact security behavior intentions. Unlike Egelman and Peer, Gratian et al. reported that rational decision-makers tend to have safe security intentions regarding device securement. We justify the reason for this difference by the fact that Gratian et al. included demographics and personality traits as an extension to Egelman and Peer’s work that led some predictors having a suppressor influence over other variables. We chose to use Egelman and Peer’s results in this work to develop the formalized users’ security-related behaviors according to specific security decisions.
Chapter 5

Modeling User Security Behavior

After reviewing studies on how the decision-making process relates to security behavior, in Chapter 4 we built the appropriate knowledge base, and selected the most relevant requirements to model in a formal manner how users interact differently with the security systems. We divided the content of this chapter into two sections: First, discussing the practical considerations that we looked at before implementing the formal model. We explain the model knowledge base, where we determine the functional entities that systems usually offer to end-users according to the scope of this research. Second, we specify the approach by which the requirements set converted into a formal model. For this research, we consider that the appropriate approach to model user’s security behavior is through the use of graph-based knowledge representation of Finite-State Automata (FSA).
5.1 Model Considerations

It is challenging to model human-machine interactions because of the complexity of human behavior and the broad set of knowledge requirements. Although we chose a specific knowledge base, it is still difficult to model and examine every aspect of a user’s security-related behavior that is relevant to device securement, password generation, proactive awareness, and updating. To model user’s behavior across the different aspects of SeBIS, we built an architecture that supports decomposition of behavior. Figure 5.1 shows the graphical representation of the knowledge base architecture that describes the structure and the organization of the components that we integrated into the model.

Figure 5.1: Knowledge base architecture
The knowledge base architecture includes different levels of abstractions, in order to ease the debugging and lessen the complexity. We decomposed the structure into different sorts of security services on multiple layers, starting from (1) as the highest level of abstraction and ending with (4) as the lowest level of abstraction. By doing so, we eliminate a fair bit of confusion around which security aspect we employed for a specific SeBIS dimension, double-check that each aspect of the model is covered, and ensure that the quality of the model is at its best by not mixing the levels of detail. Although decomposition ensured that our architecture is no longer monolithic, it ignores interactions between behaviors.

5.1.1 Levels of Abstraction

Here, we illustrate each level of abstraction in detail.

- **Layer (1)** is the most abstract of the four security service check layers. It as-similates the SeBIS concepts: device securement, password generation, proactive awareness, and updating that are specified by Egelman and Peer.

- **Layer (2)** breaks down SeBIS concepts into four security settings (i.e., device protection, update mechanism, password age, and attention to threats) either to check for possible additional sub-services or not based on the user decision.

- **Layer (3)** has a sub-tree that descends from the screen-locking feature and is enabled by what the user chooses. This level is the latter for device securement, updating, and proactive awareness.

- **Layer (4)** covers as much as possible of password generation details because passwords are the first line of defense against cybercriminals gaining access to sensitive data. Too many users underestimate password vulnerabilities by choos-
ing common passwords, ignoring simple advice about password strength, and neglecting good password practices.

5.2 Model Paradigm

The modeling paradigm was selected after looking at several possible techniques of modeling, including Markov chains and architectural representations. We decided that the most appropriate method of representing user behavior is through the use of a Finite-State Automata (FSA) because it allows us to visualize the graphical diagram of the user’s behavior easily. It enables the use of well-defined tools to perform automated analysis early in the design phase, which would empower us to reason about the logical representations of the user’s behavior at the time and to evaluate alternative design options in case there were profound implications. We developed the models that are representing our knowledge base by following the principles of Finite-State automata (FSA) [48].

In order to choose the correct platform for the purpose of designing and verifying the formal model of user’s behavior, several formalisms such as NuSMV [18], Uppaal [56], PVS [47], and Z3 [44] were considered carefully. We chose Uppaal [11][36][56], due to its ability to model timing aspects that are critical for cybersecurity, as well as its ability to generate and visualize counterexamples. Uppaal represents models as timed automata, and Uppaal formalism enables compositionality supports model checking over networked timed automata using temporal logics. This modeling paradigm allows the execution of requirements as temporal logic queries to check the satisfaction of relevant safety properties exhaustively. We next describe the timed automata formalism used by Uppaal.
5.2.1 Mathematical Representation Within the Model Paradigm.

Uppaal uses timed automata [4], a subset of hybrid automata, as a modeling formalism. One of the essential requirements in the design of human-machine interactions is to be able to model the time associated with the execution of operations or rules. A timed automaton is a finite automaton extended with a finite set of real-valued clocks. Clock or other relevant variable values used in guards on the transitions within the automaton. Based on the results of the guard evaluation, a transition may be enabled or disabled. Variables can be reset and implemented as invariants at a state. Modeling timed systems using a timed-automata approach is symbolic rather than explicit. It allows for the consideration of a finite subset of the infinite state space on-demand (i.e., using an equivalence relation that depends on the safety property and the timed automaton), which is referred to as the region automaton. There also exists a variety of tools to input and analyze timed automata and extensions, including the model checker Uppaal and Kronos [12].

- **Timed Automaton (TA)**

A timed automaton is a tuple $(L, l_0, C, A, E, I)$, where: $L$ is a set of locations; $l_0 \in L$ is the initial location; $C$ is the set of clocks; $A$ is a set of actions, co-actions, and unobservable internal actions; $E \subseteq L \times A \times B(C) \times 2^C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset; and $I : L \rightarrow B(C)$ assigns invariants to locations.

We define a clock valuation as a function $u : C \rightarrow \mathbb{R}_{\geq 0}$ from the set of clocks to the non-negative reals. Let $\mathbb{R}^C$ be the set of all clock valuations. Let $u_0(x) = 0$ for all $x \in C$. If we consider guards and invariants as the sets of clock valuations (with a slight relaxation of formalism), we can say $u \in I(l)$ means $u$ satisfies $I(l)$. 
• **Timed Automaton Semantics**

Let \((L, l_0, C, A, E, I)\) be a timed automaton \(TA\). The semantics of the \(TA\) is defined as a labelled transition system \(\langle S, s_0, \to \rangle\), where \(S \subseteq L \times \mathbb{R}^C\) is the set of states, \(s_0 = (l_0, u_0)\) is the initial state, and \(\to \subseteq S \times (\mathbb{R}_{\geq 0} \cup A) \times S\) is the transition relation such that:

1. \((l, u) \xrightarrow{d} (l, u+d)\) if \(\forall d': 0 \leq d' \leq d \Rightarrow u + d' \in I(l)\)

2. \((l, u) \xrightarrow{a} (l', u')\) if \(\exists e = (l, a, g, r, l') \in E\) such that \(u \in g, u = [r \mapsto 0] u\) and \(u' \in I(l)\)

where for \(d \in \mathbb{R}_{\geq 0}\), \(u+d\) maps each clock \(x\) in \(C\) to the value \(u(s) + d\), and \([r \mapsto 0] u\) denotes the clock valuation which maps each clock in \(r\) to 0 and agrees with \(u\) over \(C \setminus r\).

Note that a guard \(g\) of a \(TA\) is a simple condition on the clocks that enable the transition (or, edge \(e\)) from one location to another; the enabled transition is not taken unless the corresponding action \(a\) occurs. Similarly, the set of reset clocks \(r\) for the edge \(e\) specifies the clocks whose values are set to zero when the transition on edge executes. Thus, a timed automaton is a finite directed graph annotated with resets of and conditions over, non-negative real-valued clocks. Timed automata can then be composed into a network of timed automata over a common set of clocks and actions, consisting of \(n\) timed automata \(TA_i = (L_i, l_{i0}, C, A, E_i, I_i)\), \(1 \leq i \leq n\). This enables us to check reachability, safety, and liveness properties, which are expressed in temporal logic expressions, over this network of timed automata. An execution of the \(TA\), denoted by \(\text{exec}(TA)\) is the sequence of consecutive transitions, while the set of execution traces of the \(TA\) is denoted by \(\text{traces}(TA)\).
Chapter 6

Generating Linear-Time Security Properties

The set of specifications (i.e., properties), which we expect the formal model to meet, are originally generated as ideas stated in natural language, later translated into a more formal and well-defined language in order to be readable and refined by computer machines [28]. In the following sections, we first discuss the fundamental concept of linear-time properties and some significant, though relatively simple, types of such properties. Second, we describe the formal language that is used in Uppaal to encode model specifications. Uppaal uses a formal language called Timed Computation Tree Logic (TCTL). Finally, we illustrate the security properties of our knowledge base that we have generated through Uppaal to check and verify whether the design of our developed model is valid or not.
6.1 Overview of Linear-Time Properties

A linear-time property is an essential prerequisite for the traces of transition systems [7]. The linear-time property specifies the traces that one requires the transition system to exhibit. In an ideal situation, a linear time property explains the behavior of systems in the process of execution. In the next two subsections, we discuss the different aspects of linear-time behavior, deadlock, invariants, and liveness properties as related to linear-time properties. In the course of the work, we outline the basic concepts that make up the linear temporal system and discuss an in-depth of the linear-time properties.

A linear-time property is simply a $\omega$-language, set of infinite-length sequences of symbols, over $2^{\text{AP}}$ where $\text{AP}$ is a set, finite or countably infinite, of atomic propositions. Programs that are not divergent (e.g., endless loops), usually have a terminal state without outgoing transitions. When a system stops at the instance when at least one component is in a non-terminal state, the result is a deadlock scenario. In the deadlock scenario, at least one element should continue when the system has halted [7]. Typically, a deadlock scenario happens when the components in a system mutually depend or wait upon each other to progress. Mathematically, the interpretations of linear-time properties are represented as follows:

1. By letting $P$ be a linear-time property over $\text{AP}$ and $T$ be a transition system over $\text{AP}$ too in a system without terminal states, then in this case $T$ satisfies $P$, expressed as:

   $$ T \models P, \text{ when } \text{Traces}(T) \subseteq P. $$

2. Letting $\pi \in \text{Paths}(T)$, then it translates to mean that $\pi$ satisfies $P$, expressed as:

   $$ \pi \models P, \text{ when } \text{Traces}(\pi) \in P. $$
3. Letting \( s \) be a state of \( T \), it translates that \( s \) satisfies \( P \), expressed as:

\[
s \models P, \text{ when } \text{Traces}(s) \subseteq P
\]

Generally, \( P \) is satisfied by a transition system \( T \) when all the traces are in \( P \). The main idea in the above statement is that \( P \) contains all the admissible behaviors of the transition system.

In a condition where two transition systems \( T_1 \) and \( T_2 \) have a similar set of traces, then it is expected that they satisfy the same Linear-time properties. Assuming \( T_1 \) and \( T_2 \) have similar traces and let \( P \) be an arbitrary linear-time property, if in any case \( T_1 \models P \), then \( \text{Traces}(T_1) \subseteq P \), implying that \( \text{Traces}(T_2) \subseteq P \) since \( \text{Traces}(T_1) = \text{Traces}(T_2) \). When dealing with software design, it is essential to note the above expressions because if \( \text{Traces}(T_1) \subseteq \text{Traces}(T_2) \), then \( T_1 \) is a correct implementation of \( T_2 \). \( T_2 \) is considered an abstract model, while \( T_1 \) is treated as one particular implementation. Generalizing the above statement, \( T_1 \) cannot have properties that are not exhibited in \( T_2 \).

A transition system representing the computer system depends on either a state-based approach or an action-based approach. An action-based view is derived from states and meticulously relegates to the action labels [7]. The state-based approach is derived from actions, but the labels contained in the state sequences are considered — transition systems model the hardware and software systems. The verification algorithms are based on the state graph defining a transition system. Along with execution, some sequences take the form of \( L(s_0) \), \( L(s_1) \), and \( L(s_2) \) and register valid sets of atomic propositions. The sequences described above are referred to as traces. In examining the linear-time properties, we shall consider the following traces in a simplified mode.

We let the transition system \( TS = (S, Act, \rightarrow, I, AP, L) \) where all terminal states are infinite. The trace of execution \( \rho = s_0 \alpha_0 s_1 \alpha_1 \) is expressed as:
Trace(π) = L(s_0) L(s_1) . . .

The traces of a transition system are therefore infinite words over the alphabet \(2^{AP}\), expressed as: Traces (TS) \(\subseteq (2^{AP})^\omega\). The state graph of the transition system TS, notation \(G(TS)\), is the digraph \((V, E)\) with vertices “\(V = S\) and edges \(E = \{(s, s') \in S \times S \mid s' \in Post(s)\}\)” The state graph of the transition system contains a vertex for every state in the transition system and edge between each vertex \(s\) and \(s'\), where \(s'\) succeeds \(s\) in the transition system of an action \(\alpha\). The atomic propositions and transition labels are omitted to obtain a state graph for the transition system. Several transitions that contain different action labels between the states are shown by a single edge.

### 6.1.1 Invariant

An invariant refers to a linear-time property that is provided by \(\Phi\) condition for any given states and remains true for all the reachable states. A good example is described by in mutex property \(\Phi = \neg crit_1 \lor \neg crit_2\). Invariants are safety properties that define that no bad activity should happen. A typical safety property occurs where there is mutual exclusion property. That is, a bad thing having > 1 process never occurs. Deadlock freedom is another example of a typical safety property. If there is propositional logic formula \(\Phi AP\), then a linear-time property \(P_{inv}\) over \(AP\) is an invariant in the sense that: \(P_{inv} = \{A_0 A_1 A_2\. \in (2^{AP})^\omega \mid \forall j \geq 0.Aj \vdash \Phi\}\), where \(\Phi\) is referred to as the invariant condition of \(P_{inv}\).

### 6.1.2 Liveness Properties

Liveness properties compliment safety properties in the assertion that “something good will happen.” Doing nothing indicates that nothing terrible will happen, and therefore, it
approves a safety property. While finite traces represent safety violations, liveness violations are represented by infinite traces. In a wrap up about the linear-time properties, liveness property does not practically rule out finite behavior [7]. Instead, the liveness properties constrain the infinite behaviors. Every trace that refutes a safety property contains a finite prefix. Two underlying factors remain un-refuted. First, invariants refer to safety properties having bad prefix $\Phi \ast (\neg \Phi)$. Second, a safety property is only regular when the sets of wrong prefixes are a regular language.

The set of systems $TS$ of the reachable transition state is established by a search logarithm. A sequence of $(I)$ atomic preposition sets is referred to as a trace. A transition system $TS$ trace is achieved by projecting the possible paths of the sequence state labels. A linear-time property, by definition, is composed of a set of infinite words spread over $2^{AP}$.

When two transition systems have the same linear–time properties, they are trace equivalent. Purely state-based linear-time properties are also referred to as invariant, and to hold for reachable states, they require the logic formula $\Phi$ [62]. To check invariants, a depth-first search is advised. If an invariant is refuted, the depth-first search stacks provide a counter. The bad prefixes invariants exhibit what is referred to as safety properties. Invariants have finite behaviors such that the traces that refute the bad prefix that caused it. Liveliness properties in linear–time properties provide that when finite behavior is not ruled out, then finite behavior exists. All linear–time properties are equivalent to linear–time properties that have liveness and safety properties. Sometimes there exist unrealistic traces, to rule out these unrealistic traces, fairness assumptions are used [62]. The fairness assumptions consist of strong, unconditional, and weak constraints that occur with infinite executions. Determining liveness properties requires fairness assumptions, but as long as they are realized, they are irrelevant for safety properties.
6.2 Specification Language in Uppaal

The process of verification in Uppaal operates with a specific type of query language that is used to specify a set of properties that need to be examined. The query language is a subset of Computation Tree Logic (CTL) called Timed CTL (TCTL) [10]. The syntax of the Timed Computation Tree Logic is expressed as follows:

\[ \Phi ::= a \mid g \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid E (\Phi_1 \cup^J \Phi_2) \mid A (\Phi_1 \cup^J \Phi_2), \text{ where} \]

- \( a \) is an atomic action.
- \( g \) is a clock constraint.
- \( E \) means ”for some paths.”
- \( A \) means ”for all paths.”
- \( J \) is an interval whose bounds are natural number.

TCTL is similar to CTL in having temporal connectives that are expressed as pairs of symbols. Such that, the first element of the pair represents one of the path quantifiers that is either \( A \) or \( E \) whereas the second element of the pair is one of the state quantifiers that is one of the following:

- \( G \) means ”all states in a path.”
- \( F \) means ”some state in a path.”

In Table 6.1 from [56], we can see the different combinations of path formulae and state formulae accepted by Uppaal. Figures 6.1 to 6.4 represent the computation trees of the formulae.
### Table 6.1: Temporal connectives in Uppaal

| Symbol | Name               | Property in Uppaal |
|--------|--------------------|--------------------|
| AG     | invariantly        | A[ ]              |
| EG     | potentially always | E[ ]              |
| AF     | eventually         | A<>               |
| EF     | possibly           | E<>               |

![AG computation tree](image)

**Figure 6.1: AG computation tree**

![EG computation tree](image)

**Figure 6.2: EG computation tree**
Figure 6.3: AF computation tree

Figure 6.4: EF computation tree
6.3 Knowledge Base Properties

After gaining ground comprehension of how Uppaal specifies properties as Timed CTL formulas using four temporal operators only (i.e., $A, E, G, F$), we decided to establish our specific properties concerning each aspect of the knowledge base. Sections 6.3.1 to 6.3.4 will present a further illustration of why and how we chose the most critical matters within different security settings that are parts of device securement, password generation, proactive awareness, and updating. Based on existing literature, we set up examples for the different settings to separate good decisions from poor ones. Organizations can also replace our examples of good or bad security decisions according to their own policy decisions. The use of such properties enable us to examine users’ behaviors and distinguish which user exhibits poor security-related practices. Our method allows policy makers to set their own policy threshold for “bad” decisions. They do not need to use the choices in our examples.

6.3.1 Device Securement

- Device Protection

  We chose device protection as the first criterion for device securement. It means whether or not users protect their devices by passwords, PIN codes, fingerprints, or patterns. Even though locking devices of all kinds is a simple security task, it is sometimes undervalued by end-users. In the Pew Research Center survey, conducted in 2017, 28% of American mobile phone users reported that they do not use PIN codes or any other security feature to access their smartphones [46]. This matter of security is a lot more critical in the work environment. Because; for instance, if some employees are working outside the workplace using
portable devices (e.g., laptops, tablets, or smartphones) as their primary work computer, they could leave the device unprotected in some places as in a hotel room or a car. By doing so, if their device is stolen, it is already unlocked, and the company’s data would be in the hand of an unauthorized individual. Most organizations require this control for devices holding their data, but it may be hard to monitor and enforce across all ecosystems and devices.

- **Screen-Locking**

  Password-protected screen saver feature is about setting up the device to lock off automatically after some time of inactivity. Some end-users find it a little troublesome thing to do when they have to consistently login again every moment the timeout is exceeded. Others have a low perception of the threat; they believe nothing would go wrong since they are around their portable devices almost all the time, especially smartphones [1]. Some others do set a password-protected screen saver, but they adjust the default timeout time (i.e., often 15 minutes) to a much longer time [55]. We examine this side of device securement because leaving the device without a password-protected screen saver would allow malicious individuals (e.g., insider threats) to access data or perform some tasks they are not entitled to see or do [15]. Insider threat is one of the most difficult security issues; malicious insiders can put the organization at a greater risk than outsiders because they are more familiar with security infrastructure, practices, and vulnerabilities. They can more easily avoid detection and remain hidden for a long period of time.

  In Table 6.2, we can see how we translate the criteria mentioned above into Timed CTL formulate whereas Figure 6.5 depicts device protection, screen-locking, and screen-locking timeout state-transition graphs, respectively.
Table 6.2: Device securement properties

| No. | Knowledge Base Property          |
|-----|----------------------------------|
| 1   | E<> (device_protection.enabled)  |
| 2   | E<> (device_protection.disabled) |
| 3   | E<> (screen_locking.turned_on)   |
| 4   | E<> (screen_locking.turned_off)  |
| 5   | E<> (timeout.short)              |
| 6   | E<> (timeout.long)               |

Figure 6.5: Device securement state-transition graphs
6.3.2 Password Generation

- **Password Age**
  
The problem of passwords is a perennial in cybersecurity. Much advice is given, and policies are enforced, but still the problem of weak passwords is constantly growing. Setting a maximum password age is one of the traditional techniques for maintaining proper password hygiene. It requires users to change their passwords in a periodic manner, typically between 30 and 90 days [8]. Habib et al. (2018) state that some works of study have implied that forcing users to change their password repeatedly does not lead to better security. Their study shows that 82% of the participants coincided that this technique had a positive influence on password security, and accordingly would reflect some significant reduce in password-guessing attacks [29]. Some might argue that scheduled changes make the passwords harder for users to remember, and thus more likely to be stored insecurely. Users are adjusting and developing coping mechanisms to overcome this burden by making some alterations to their current passwords. We included this rule to our set of specifications to show an example of the variety of criteria that can be included in the knowledge base.

- **Password Length**
  
People want to protect their personal information, but they are not willing to pay a little more effort for it. The biggest example of this is that despite the continuous security warnings that alert users about the use of weak passwords and the serious consequences that result from such behavior, they still use short and easy-to-remember passwords. According to the Psychology of the Password Report, 47% of people who participate in the study declared that they prefer to
choose easy passwords because they are afraid of forgetting them [37]. Martin et al. (2012) illustrate that guessing a password through a brute-force attack is most likely unsuccessful against long and complicated passwords that contain capital and small letters, digits, and symbols. They mention that cracking a complicated password with lengths of 4, 8, or 16 characters would take approximately 81 seconds, 210 years, and 1.4 quintillion years, respectively [41]. According to Maddox and Moschetto (2019), when users assign a password for some account, they should maintain an adequate length, avoid well-known character substitutions, and use a variety of letters, numbers, and special characters [40]. We thought that the length of a password is necessary to look at to identify those whose accounts are vulnerable to password attacks.

- **Password Re-usability**

  With so many accounts to handle and keep track of, it can be tempting for users to use one password across them all and bring themselves some peace of mind; users might live to regret it. Re-using the same password for multiple sites carries more risk than writing down separate passwords on a piece of paper. If individuals decided to assign one password for their different accounts, they would allow attackers to compromise other accounts that use the same password [33]. The struggle with password protection is that the majority of end-users acknowledge the risk and the consequences of password re-use; yet, 35% of them ignore this knowledge in favor of remembering their easy and familiar passwords [37]. We chose to cover this aspect of password generation because, in the worst-case scenario, the resulting risk is not limited to the end-users but extends to the workplaces and coworkers if they re-use the work password for some other personal account.
Table 6.3 lists the equivalent expressions of password age, length, and reusability in Uppaal specification language. In Figure 6.6, we can notice the state-transition graphs of password generation aspects.

Table 6.3: Password generation properties

| No. | Knowledge Base Property                               |
|-----|-------------------------------------------------------|
| 1   | E<> (password_age.non_expired)                        |
| 2   | E<> (password_age.expired)                            |
| 3   | E<> (password_length.long)                            |
| 4   | E<> (password_length.short)                           |
| 5   | E<> (password_reusability.unused)                     |
| 6   | E<> (password_reusability.used)                       |
6.3.3 Proactive Awareness

- **Spot Signs of Threat**

There is no doubt that the interest in cybersecurity is growing and expanding, making people more educated and cautious about online information sharing, fake e-commerce sites, scams, and security threats. There is always a “but” in this imperfect world because there is a significant number of people who lack digital security awareness and education, which in turn affect their ability to recognize threats even if there were apparent signs. End-users would not be able to protect themselves from identity theft if they were unable to spot a sign of spyware on their device, recognize a social engineering attempt, identify phishing or spoofing email, or any other elusive activities. According to Verizon’s Data Breach Investigations Report (DBIR) (2019), the phishing attack was one of the leading causes of data breaches. It was a contributing factor in 32% of confirmed data exposures, and 78% of cyber-espionage incidents [58]. On account of this, we stressed the importance of spotting early warning signs by investigating the users’ ability to recognize and avoid phishing scams before it is too late.
• **Report Threat**

In “If You See Something, Say Something®” national campaign, which raises public awareness of the indicators of terrorism-related crime, we are encouraged to report the authorities if something does not seem quite right to keep ourselves and our communities safe [45]. It is exactly the case in organizations’ environments; reporting possible security incidents can save valuable crucial time in the early stages of breach detection [21]. If employees know about cyberattack types and how they look and occur, they are more likely to notice unauthorized changes that have taken place on their systems. They probably would be more confident and willing to reach out to the IT department. Lack of cybersecurity awareness, training, and vigilance, and miscommunication between employees and the IT team would make the former hesitate and question themselves whether they have caught something real or not. In this research, this condition is enabled by the previous security aspect since users would not report something they did not notice in the first place. The idea of including this area of interest to our research is to determine whether users are truly not educated enough, or they are just too reckless to report such urgent matter.

• **Policy Compliance**

Identifying, determining, and handling risks to the confidentiality, integrity, and availability of an organizations’ assets is a top-notch priority. Security policies and procedures are part of the hierarchy of any organizations’ management control to maintain the security of sensitive data, the most critical asset, from the complex and ever-evolving threat landscape. One of the biggest concerns for any organization is how to protect data from its employees [2]. Security policies and guidelines are put in place to draw a line for employees between what is accept-
able and unacceptable to do when they interact with the information system. The problem lies in the employees’ non-compliance [49]. According to Mutlaq et al. (2016), non-compliant behavior can be categorized as: first, deviant behavior that is driven by intentional or planned desire to harm the organization entity. Second, negligent behavior that is intended to go against security policy but with no malicious intent to cause harm. Third, ignorant behavior that stems from unawareness due to a lack of cybersecurity knowledge and training. With these different interpretations of security policy being violated, we were eager to look into end-users’ adherence to security policy it eases the process of identifying the right policy actions to impose against a specific user.

Table 6.4 shows the linear-time properties that have been generated to investigate the knowledge base aspect of proactive awareness. We can see in Figure 6.7 the graphical representation of proactive awareness criteria.

| No. | Knowledge Base Property                      |
|-----|---------------------------------------------|
| 1   | E<> (spot_signs.yes)                        |
| 2   | E<> (spot_signs.no)                         |
| 3   | E<> (report_threat.yes)                     |
| 4   | E<> (report_threat.no)                      |
| 5   | E<> (security_policy.complied)              |
| 6   | E<> (security_policy.violated)              |
Figure 6.7: Proactive awareness state-transition graphs
6.3.4 Updating

- Updating Mechanism

Security-related software updates are one of the single most important security protection tools that end-users should pay more attention to and ensure that they are being installed on a regular periodic schedule. There are several options for the operating system and application updates that end-users can configure, such as how updates are downloaded and installed (i.e., automatically or manually) on their devices. Most modern software systems are set to download and install security and other essential updates automatically without the need for humans to make decisions. In contrast, the manual update allows users to gain better control over their devices by choosing which update to install and when. There is no certain position where we can say that one mechanism is better and safer than the other because it all depends on users’ behaviors regarding this matter. It may come to mind that the automatic mechanism appears as a more responsible selection. Still, the manual update is responsible as well in case end-users ensure that their systems are up-to-date as soon as a new update is available. We chose including users’ preferences of the updating mechanism to observe their behaviors from different angles with respect to their differences.

- Time to Update

While some software updates and patches are released to address security bugs that have been discovered in previously installed software, some end-users avoid or delay the installation because they consider that these incremental updates are just useless technical additions [57]. Herein lies the risk for those users who download and install updates manually after they were available a while ago. Users
always encouraged to apply security-related updates for their operating systems and important applications as soon as possible after those updates are released. The reason for this is that after vulnerabilities are disclosed, it is only a matter of time, sometimes not much time at all, before cybercriminals use that information to create exploits. We discern from this that delaying the installations of the latest updates and patches forms windows of opportunities for malicious individuals to exploit open security vulnerabilities on the users’ devices [52]. We investigate how long it takes end-users to apply available updates to identify those how are negligent and pose the greatest risk to cybersecurity systems.

- **Time To Reboot**

  Software developers have unremittingly endeavored to improve security by excluding the user role from the software update cycle. They found that user intervention remains a must because some updates require a device reboot to allow changes to take effect [59]. There are operating systems, such as Microsoft Windows, developed to alert users if a reboot is required after updates are installed. In this case, the system shows up a notification pop-up that a reboot will occur within a while (i.e., usually 10 minutes). Users are given the option either to reboot the device immediately or to postpone for an additional specific time. If users chose to postpone, the warning dialog would appear again with the same options [59]. Users delay the rebooting task because they might have some pressing matters that keep them from rebooting for a couple of hours. The decision to postpone rebooting more than once could negatively impact the security of the device because, for the computer system, the update installation is not completed. We observed this aspect of updating to draw the attention of negligent and unaware users to the importance of immediate reboot if required.
In Table 6.5, we present the formal specifications of updating that are expressed in terms of timed temporal logic TCTL while Figure 6.8 visualizes updating mechanism, time to update, and time to reboot as Finite-State Automata (FSA).

Table 6.5: Updating properties

| No. | Knowledge Base Property                           |
|-----|--------------------------------------------------|
| 1   | E<> (updating_mechanism.automatically)           |
| 2   | E<> (updating_mechanism.manually)               |
| 3   | E<> (time_to_update.short)                      |
| 4   | E<> (time_to_update.long)                       |
| 5   | E<> (time_to_reboot.right_away)                 |
| 6   | E<> (time_to_reboot.after_ahile)                |
Figure 6.8: Updating state-transition graphs
Chapter 7

Generating User-Specific Security Policy

This chapter combines and builds upon all previous chapters. By examining end-users’ decisions, we would be able to identify any possible weaknesses that exist in users’ security-related behaviors. We further can generate the proper policy that would close any security gap that results from lack of awareness and poor practices. In this chapter, we generate and model several test cases with different security behaviors, concerning device securement, password generation, proactive awareness, and updating, as Finite-State Automaton (FSA). For each separate test case, we generate a set of user-specific linear-time properties that we use later to meticulously verify the formal behavioral models by using the reachability analysis technique. In our case, this technique contributes to automatically distinguishing the good security behaviors from the bad ones. Moving a step closer to achieving our purpose, we elucidate our method and generate the appropriate policy, which best meets a particular user’s security needs.
7.1 Automated Analysis

We seek to analyze users’ behaviors in order to make a careful analysis and draw out their poor security decisions that have a significant impact on the security system. To ultimately achieve our goal, we designed six test cases that cover as much as possible of different security behaviors exhibited by users in real-world scenarios that revolve around device securement, password generation, proactive awareness, and updating. For each test case, we represent the user’s behavior as a state-transition graph using the Uppaal tool\(^1\), manually generate user-specific linear-time properties, apply reachability analysis, and generate user-specific policy. The first three steps fall within the fourth stage of our User-Specific Policy Generation Framework (USPGF). In this section, we demonstrate one example of these test cases as an illustration of our formal method-based approach.

7.1.1 Representing Test Cases

In order to have reliable test cases, we had to make several assumptions and predictions of how some users might behave and make security-related decisions. In one test case, we created a scenario with a user named Tom, who works as a Data Entry Specialist at a network marketing company. Tom is assigned a laptop to perform duties directly related to the business of the company and to allow him to work remotely and outside of regular working hours. On this basis, the company requires him to be responsible and take reasonable precautions to protect and maintain the laptop and its content. For this research, we are focusing on capturing Tom’s security behavior rather than his system

\(^1\)integrated tool environment for modeling, validation, and verification of Finite-State Automaton (FSA)[56]
role. Figure 7.1 represents the state-transition graph of Tom’s security-related behavior.

![Tom’s state-transition graph](image)

**Figure 7.1: Tom’s state-transition graph**

### 7.1.2 Generating User-Specific Security Properties

In the previous section, we have successfully managed to draw the user’s security-related behavior through the modeling graphs of Finite-State Automata (FSA). We made Tom deliberately exhibit different good and poor security behaviors regarding device security, password generation, proactive awareness, and updating. In order to automatically analyze Tom’s behavior, we needed to create linear-time properties manually, which are generated specifically for Tom. In Table 7.1, we combined the resulted properties with knowledge base properties that we previously specified in Section 6.3, Chapter 6. We are generating these properties in order to distinguish the aspects where Tom has failed to apply proper security practices.
7.1.3 Rechability Analysis

Reachability analysis or model checking is a verification procedure for models that are designed based on the state-transition concept. According to Kong et al. (2015), reachability analysis is a technique used in a state-transition system in order to find out the type and number of states which can be accessed through a particular system model [35]. Reachability analysis allows formal analysis for validation, verification, and performance, explained as follows:

- **Validation**: A simulation of the process is shown where the model should reflect what it intended to represent.

---

Table 7.1: Knowledge base x Tom’s security properties

| No. | Security Linear-Time Property |
|-----|--------------------------------|
| 1   | E<> (Tom.device_protection and device_protection.enabled) |
| 2   | E<> (Tom.device_protection and device_protection.disabled) |
| 3   | E<> (Tom.screen_locking and screen_locking.tarded_on) |
| 4   | E<> (Tom.screen_locking and screen_locking.tarded_off) |
| 5   | E<> (Tom.timeout and timeout.short) |
| 6   | E<> (Tom.timeout and timeout.long) |
| 7   | E<> (Tom.updating_mechanism and updating_mechanism.manually) |
| 8   | E<> (Tom.updating_mechanism and updating_mechanism.automatically) |
| 9   | E<> (Tom.time_to_update and time_to_update.long) |
| 10  | E<> (Tom.time_to_update and time_to_update.short) |
| 11  | E<> (Tom.time_to_reboot and time_to_reboot.after_awarewhile) |
| 12  | E<> (Tom.time_to_reboot and time_to_reboot.right_aware) |
| 13  | E<> (Tom.password_age and password_age.non_expired) |
| 14  | E<> (Tom.password_age and password_age.expired) |
| 15  | E<> (Tom.password_length and password_length.long) |
| 16  | E<> (Tom.password_length and password_length.short) |
| 17  | E<> (Tom.password_reusability and password_reusability.unused) |
| 18  | E<> (Tom.password_reusability and password_reusability.used) |
| 19  | E<> (Tom.spot_signs and spot_signs.no) |
| 20  | E<> (Tom.spot_signs and spot_signs.yes) |
| 21  | E<> (Tom.report.threat and report.threat.no) |
| 22  | E<> (Tom.report.threat and report.threat.yes) |
| 23  | E<> (Tom.security.policy and security_policy.violated) |
| 24  | E<> (Tom.security.policy and security_policy.complied) |
- **Verification**: A checking process is carried out to ensure that the specifications meet the model that is built.

- **Performance**: A set of predictions about the key performance indicators is made.

Among these procedures, we are only verifying whether our specifications (i.e., properties) holds in any, some, or all state of user’s behavior model. According to Eleftherakis et al. (2001), “A model checker takes a model and a property as inputs and outputs either a claim that the property is true or a counterexample falsifying the property.” [24]. Figure 7.2 shows the model checking approach that we used to check if our model conforms to its specifications or not [7]. In this research, we favored reachability analysis over other methods such as graph matching approach because it provides the opportunity to intensely and automatically check all possible paths in which security properties might not be satisfied. Reachability analysis helps with determining the general drawbacks while designing the formal model, and validating that the basic behavior of the model is as expected.
For the set of combined linear-time security properties, we applied reachability analysis using Uppaal to see which properties were satisfied and which were not. The results of this procedure, shown in Table 7.1, allow us to examine Tom’s behavior and identify his security weaknesses. In Figure 7.3 and 7.4, we can see two examples of reachability analysis results in Uppaal, where Tom enabled manual updating and did not install the new updates within the first day of their release.

Uppaal executes reachability analysis and checks whether a state is reachable either in a forward or backward manner, and both can make use of Breadth-First Search (BFS) or Depth-First Search (DFS) algorithms for traversing graphs, respectively. We chose Breadth-First Search (BFS) to verify our reachability properties (i.e., security properties) in users’ state-transition graphs because it allows traversing a graph with ease. It finds the shortest paths to a destination state in the sense of fewest edges. Also, Breadth-
First Search (BFS) is a complete algorithm that ends within finite time and finds the destination state if it is present, even in infinite graph problems. It makes Breadth-First Search (BFS) more advantageous than Depth-First Search (DFS) algorithm that might get lost in an infinite loop and would not find a path to the destination state, even if it exists.

Table 7.1: Tom’s reachability analysis results

| No. | Property Status |
|-----|-----------------|
| 1   | satisfied       |
| 2   | not satisfied   |
| 3   | satisfied       |
| 4   | not satisfied   |
| 5   | satisfied       |
| 6   | not satisfied   |
| 7   | satisfied       |
| 8   | not satisfied   |
| 9   | satisfied       |
| 10  | not satisfied   |
| 11  | satisfied       |
| 12  | not satisfied   |
| 13  | satisfied       |
| 14  | not satisfied   |
| 15  | satisfied       |
| 16  | not satisfied   |
| 17  | satisfied       |
| 18  | not satisfied   |
| 19  | satisfied       |
| 20  | not satisfied   |
| 21  | satisfied       |
| 22  | not satisfied   |
| 23  | satisfied       |
| 24  | not satisfied   |
Figure 7.3: Example (1) of Tom’s reachability analysis results

Figure 7.4: Example (2) of Tom’s reachability analysis results
### 7.2 Generating User-Specific Policy

In this section, we generate and define a set of policies to be imposed on specific users who exhibited poor security practices. The type of policies to be enforced depends on the users’ decisions that represent their adherence to the rules set, which we established for each security aspect highlighted in this research: Device Securement (DS), Password Generation (PG), Proactive Awareness (PA), and Updating (U). We assign each security aspect one of three standardized policy types that we drafted as follows:

1. **Strict Policy** ($s$): Applied when the user is exhibiting severe disregarding for the appropriate security measures.

2. **Moderate Policy** ($m$): Applied when the user is exhibiting somewhat irresponsibility in following the appropriate security measures.

3. **Restricted Policy** ($r$): Applied when the user is exhibiting good security behavior with full respect to the appropriate security measures.

After analyzing users’ security-related behaviors and determining the appropriate policy for each security aspect, we generate one general security policy for each user. This general policy is composed of four policies that have been assigned previously for device securement, password generation, proactive awareness, and updating. We formulated the resulted general policy in the following manner:

$$Policy_u = \{DS(G/B), PG(G/B), PA(G/B), U(G/B)\}$$

$$= \langle P_{DS}, P_{PG}, P_{PA}, P_{U} \rangle$$
Where:

- $u$ is the user under test.
- $G/B$ indicates the behavior (i.e., Good or Bad).
- $t$ represents the type of policy to be imposed.

As per the formal description of the general policy, we can determine the appropriate policy that is suitable for Tom to be as follows:

$$Policy_{Tom} = \{DS(G), PG(G), PA(B), U(B)\} = \langle P_{DSr}, P_{PGr}, P_{PAs}, P_{Um} \rangle$$
Chapter 8

Conclusion

We were able to address the concept of Zero Trust and eliminate the trust that all users act responsibly. Most importantly, we achieved some success in answering the research question “How to automatically and satisfactorily generate fair and explicit security policy after observing and analyzing security behaviors, especially security-related decisions, exhibited by end-users in an environment with Zero Trust assumptions?” Correctly, in response to the three-sub questions of the research question, we can answer:

Q1: Is it possible to model end-user behavior using Finite-State Automaton (FSA)?

Yes, Finite-State Automaton provided the opportunity to model user behavior. It allowed showing how a user could transition from safe to unsafe state based on making some specific decisions.

Q2: Is it possible to generate linear-time security properties and then verify the reachability of good and poor behaviors?

Yes, TCTL language used to generate linear-time properties, and thus, a reachability analysis is enabled and done through Breadth-First Search algorithm (BFS).
Q3: Is it possible to compute the type of security policy that should be imposed on a specific user using the developed formal method that results from 1 and 2?

Yes, after observing security behavior and analyzing security behavior, the appropriate policy was assigned to address security gaps caused by specific user.

We designed our models as a deterministic. The deterministic approach that we applied to model users’ security-related behaviors means that we specified the inputs of knowledge base variables, which means that we expected how users behave and already know where their security weaknesses lie. Our designing approach comes at the cost where realism is almost non-existent. The analysis technique that we have performed is as close as possible to be static in which we examined user-machine interactions and performed the verification process within a non-run-time environment. Our future work will consider many different pieces of research, adjustments, and examinations that we had to left for extensive future investigations due to lack of time. Forthcoming research would involve conducting surveys, expanding knowledge base, and gathering more realistic data from information technology personnel and cybersecurity experts. We would apply more in-depth and more dynamic analysis to allow examining end-users and their security decisions in a real-time environment and monitoring their security awareness and their behaviors over time. This will enable us to check if any previously imposed policy needs some adjusting or revising.
Bibliography

[1] Y. Albayram, M. M. H. Khan, T. Jensen, and N. Nguyen. “...better to use a lock screen than to worry about saving a few seconds of time”: Effect of fear appeal in the context of smartphone locking behavior”. In *Proceedings of the Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 49–63. Berkeley, CA: USENIX, 2017.

[2] M. Alotaibi, S. Furnell, and N. Clarke. Information security policies: A review of challenges and influencing factors. In *11th International Conference for Internet Technology and Secured Transactions (ICITST)*. New York, NY: IEEE, 2016. doi: https://doi.org/10.1109/ICITST.2016.7856729.

[3] I. Alseadoon, M. F. I. Othman, and T. Chan. *Advanced Computer and Communication Engineering Technology*, chapter What Is the Influence of Users’ Characteristics on Their Ability to Detect Phishing Emails?, pages 949–962. Cham, Switzerland: Springer, 2015. doi: https://doi.org/10.1007/978-3-319-07674-4_89.

[4] R. Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1999.
[5] S. Babenyshev and V. Rybakov. Linear temporal logic LTL: basis for admissible rules. *Journal of Logic and Computation*, 21(2):157–177, 2010. doi: https://doi.org/10.1093/logcom/exq020.

[6] S. Babenyshev and V. Rybakov. Unification in linear temporal logic LTL. *Annals of Pure and Applied Logic*, 162(12):991–1000, 2011. doi: https://doi.org/10.1016/j.apal.2011.06.004.

[7] C. Baier and J. P. Katoen. *Principles of model checking*. Cambridge, MA: MIT Press, 2008.

[8] D. Barrett, K. K. Hausman, and M. Weiss. *CompTIA Security+ SY0-401 Exam Cram*, chapter Account Management, pages 422–437. Hoboken, NJ: Pearson IT Certification, 2015.

[9] R. F. Baumeister and K. D. Vohs. *Encyclopedia of social psychology*, volume 1. Thousand Oaks, CA: SAGE Publications, 2007.

[10] G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal 4.0, 2006.

[11] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Uppaal: A tool suite for automatic verification of real-time systems. *Theoretical Computer Science*, 1996.

[12] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: a model-checking tool for real-time systems. In *Proceedings of the 10th International Conference on Computer Aided Verification (CAV’98)*, volume 1998, page 546–550. Berlin: Heidelberg: Springer-Verlag, 1998.
[13] A. Buluç, S. Beamer, K. Madduri, K. Asanovic, and D. Patterson. Distributed-memory breadth-first search on massive graphs. *arXiv*, 2017.

[14] D. Canali, L. Bilge, and D. Balzarotti. On the effectiveness of risk prediction based on users browsing behavior. In *ASIA CCS ’14 Proceedings of the 9th ACM symposium on Information, Computer and Security*, pages 171–182. New York, NY: ACM, 2014. doi: https://doi.org/10.1145/2590296.2590347.

[15] D. Cappelli, A. Moore, T. J. Shimealland, and R. Trzeciak. Common sense guide to prevention and detection of insider threats, 2006.

[16] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Guaranteed error bounds on approximate model abstractions through reachability analysis. In *15th International Conference on Quantitative Evaluation of Systems (QEST)*, pages 104–121. Cham, Switzerland: Springer, 2018. doi: https://doi.org/10.1007/978-3-319-99154-2_7.

[17] J. H. Cho, H. Cam, and A. Oltramari. Effect of personality traits on trust and risk to phishing vulnerability: Modeling and analysis. In *2016 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pages 7–13. New York, NY: IEEE, 2016. doi: http://dx.doi.org/10.1109/COGSIMA.2016.7497779.

[18] A. Cimatti, E. Clarke, Giunchiglia E., Giunchiglia F., Pistore M., Roveri M., Sebastiani R., and Tacchella A. NuSMV 2: An opensource tool for symbolic model checking. In *CAV ’02 Proceedings of the 14th International Conference on Computer Aided Verification*, pages 359–364. Berlin, Heidelberg: Springer, 2002. https://doi.org/10.1007/3-540-45657-0_29.
[19] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004. doi: https://doi.org/10.1109/TPAMI.2004.75.

[20] F. Dinuzzo. Kernels for linear time invariant system identification. *SIAM Journal on Control and Optimization*, 53(5):3299–3317, 2015. doi: https://doi.org/10.1137/130920319.

[21] N. Easen. Speed is key in tackling data breach fallout. *Raconteur: Cybersecurity*, 2019.

[22] S. Egelman and E. Peer. Predicting privacy and security attitudes. *ACM SIGCAS Computers and Society*, 45(1):22–28, 2015. doi: https://doi.org/10.1145/2738210.2738215.

[23] S. Egelman and E. Peer. Scaling the security wall: Developing a security behavior intentions scale (SeBIS). In *CHI ’15 Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2873–2882. New York, NY: ACM, 2015. doi: http://dx.doi.org/10.1002/andp.19053221004.

[24] G. Eleftherakis and P. Kefalas. *Advances in signal processing and computer technologies*, chapter Towards model checking of finite state machines extended with memory through refinement, pages 321–326. World Scientific and Engineering Society Press, 2001.
[25] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In 28th International Conference on Computer Aided Verification (CAV), pages 531–538. Cham, Switzerland: Springer, 2016. doi: https://doi.org/10.1007/978-3-319-41528-4_29.

[26] G. D. Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In IJCAI ’13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pages 854–860. Palo Alto, CA: AAAI Press, 2013.

[27] M. Gratian, S. Bandi, M. Cukier, J. Dykstra, and A. Ginther. Correlating human traits and cyber security behavior intentions. Computers and Security, 73:345–358, 2018. doi: https://doi.org/10.1016/j.cose.2017.11.015.

[28] E. Gunter. From natural language to linear temporal logic: difficulties of capturing natural language specifications in formal languages for automatic analysis.

[29] H. Habib, P. Emami-Naeini, S. Devlin, M. Oates, C. Swoopes, L. Bauer, N. Christin, and L. F. Cranor. User behaviors and attitudes under password expiration policies. In SOUPS ’18 Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security, pages 13–30. USENIX, 2018.

[30] T. Halevi, J. Lewis, and N. Memon. A pilot study of cyber security and privacy related behavior and personality traits. In WWW ’13 Companion Proceedings of the 22nd International Conference on World Wide Web, pages 737–744. New York, NY: ACM, 2013. doi: https://doi.org/10.1145/2487788.2488034.
[31] T. Halevi, N. Memon, J. Lewis, P. Kumaraguru, S. Arora, N. Dagar, F. Aloul, and J. Chen. Cultural and psychological factors in cyber-security. In *iiWAS ’16 Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, pages 318–324. New York, NY: ACM, 2016. doi: https://doi.org/10.1145/3011141.3011165.

[32] C. Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *NSPW ’09 Proceedings of the 2009 workshop on New security paradigms workshop*, pages 133–144. New York, NY: ACM, 2009. doi: https://doi.org/10.1145/1719030.1719050.

[33] B. Ives, K. R. Walsh, and H. Schneider. The domino effect of password reuse. *Human-computer etiquette*, 47(4):75–78, 2004. doi: https://doi.org/10.1145/975817.975820.

[34] J. Kindervag, E. Ferrara, R. Hollandand, and H. Shey. Developing a framework to improve critical infrastructure cybersecurity. Technical report, Forrester Research, Inc, 2013.

[35] S. Kong, S. Gao, W. Chen, and E. Clarke. dReach: \(\delta\)-reachability analysis for hybrid systems. In *21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 200–205. Berlin/Heidelberg: Springer, 2015. doi: https://doi.org/10.1007/978-3-662-46681-0_15.

[36] K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *Proc. of Fundamentals of Computation Theory*, number 965 in Lecture Notes in Computer Science, pages 62–88, 1995.
[37] LastPass. Lastpass psychology of the password. Technical report, LastPass, 2016.

[38] H. Liu, H. H. Huang, and Y. Hu. iBFS: Concurrent breadth-first search on GPUs. In *International Conference on Management of Data (SIGMOD)*, pages 403–416. San Francisco, CA: ACM, 2016. doi: https://doi.org/10.1145/2882903.2882959.

[39] Z. Y. Liu, H. Qiao, and L. Xu. An extended path following algorithm for graph-matching problem. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1451–1456, 2012. doi: https://doi.org/10.1109/TPAMI.2012.45.

[40] I. Maddox and K. Moschetto. Modern password security for users: User-focused recommendations for creating and storing passwords, 2019.

[41] S. Martin and M. Tokutomi. Password cracking, 2012.

[42] K. D. Mitnick, W. L. Simon, and S. Wozniak. *The Art of Deception: Controlling the Human Element of Security*. Hoboken, NJ: Wiley, 2002.

[43] T. Mitsuishi, J. Suzuki, Y. Hayashi, M. Kan, and H. Amano. Breadth first search on cost-efficient multi-GPU systems. *Computer Architecture News (SIGARCH)*, 43(4):58–63, 2015. doi: https://doi.org/10.1145/2927964.2927975.

[44] L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS’08/ETAPS’08*, pages 337–340. Berlin: Heidelberg: Springer-Verlag, 2008. http://dl.acm.org/citation.cfm?id=1792734.1792766.
[45] U.S. Department of Homeland Security. If you see something, say something, 2015.

[46] K. Olmstead and A. Smith. Americans and cybersecurity. Technical report, Paw Research Center, 2017.

[47] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. Pvs: Combining specification, proof checking, and model checking. In 1996 Proceedings of Computer Aided Verification: 8th International Conference, pages 411–414. Berlin, Heidelberg: Springer, 1996. https://doi.org/10.1007/3-540-61474-5_91.

[48] Chu P. P. RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, chapter Finite State Machine: Principle and Practice, pages 313–371. Hoboken, NJ: Wiley, 2006. doi: https://doi.org/10.1002/0471786411.ch10.

[49] S. Pahnila, M. Siponen, and A. Mahmood. Employees’ behavior towards is security policy compliance. In 2007 40th Annual Hawaii International Conference on System Sciences (HICSS’07). New York, NY: IEEE, 2007. doi: https://doi.org/10.1109/HICSS.2007.206.

[50] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision computing, 27(7):950–959, 2009. doi: https://doi.org/10.1016/j.imavis.2008.04.004.

[51] W. Ruan, X. Huang, and M. Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In 27th International Joint Conference on Artificial Intelligence (IJCAI), pages 2651–2659. Stockholm, Sweden: AAAI Press, 2018.
[52] A. Sarabi, Z. Zhu, C. Xiao, M. Liu, and T. Dumitras. Patch me if you can: A study on the effects of individual user behavior on the end-host vulnerability state. In *18th International Conference on Passive and Active Network Measurement (PAM)*, pages 113–125. Cham, Switzerland: Springer, 2017. doi: https://doi.org/10.1007/978-3-319-54328-4_9.

[53] IBM Security. Cost of a data breach report. Technical report, Ponemon Institute, 2019.

[54] Team Snapchat. An apology to our employee. Web Page, February 2016.

[55] Microsoft Support. How to change the logon screen saver in windows, 2018.

[56] Uppaal. Uppaal website, 2010. http://www.uppaal.org.

[57] K. E. Vaniea, E. Rader, and R. Wash. Betrayed by updates: how negative experiences affect future security. In *CHI '14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2671–2674. New York, NY: ACM, 2014. doi: https://doi.org/10.1145/2556288.2557275.

[58] Verizon. Data breache investigations report. Technical report, Verizon, 2019.

[59] R. Wash, E. Rader, K. Vaniea, and M. Rizor. Out of the loop: How automated software updates cause unintended security consequences. In *10th Symposium On Usable Privacy and Security (SOUPS)*. Berkeley, CA: USENIX, 2014.

[60] R. West. The psychology of security. *The psychology of security: why do good users make bad decisions?*, 51(4):34–40, 2008. doi: https://doi.org/10.1145/1330311.1330320.
[61] R. West, C. Mayhorn, J. Hardee, and J. Mendel. Social and Human Elements of Information Security: Emerging Trends and Countermeasures, chapter The Weakest Link: A Psychological Perspective on Why Users Make Poor Security Decisions, pages 43–60. Hershey, PA: IGI Global, 2009. doi: http://dx.doi.org/10.4018/978-1-60566-036-3.ch004.

[62] A. Wijs. International Conference on Computer Aided Verification, chapter BFS-Based Model Checking of Linear-Time Properties with an Application on GPUs, pages 472–493. Cham, Switzerland: Springer, 2016. doi: https://doi.org/10.1007/978-3-319-41540-6_26.

[63] E. M. Wolff, U. Topcu, and R. M. Murray. Optimization-based trajectory generation with linear temporal logic specifications. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 5319–5325. New York, NY: IEEE, 2014. doi: https://doi.org/10.1109/ICRA.2014.6907641.

[64] F. Zhou and F. De la Torre. Factorized graph matching. CVPR '12 Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 127–134, 2012.
### Appendix

Table 8.1: Knowledge base + John's security properties

| No. | Security Linear-Time Property |
|-----|--------------------------------|
| 1   | E<> (John.device_protection and device_protection.enabled) |
| 2   | E<> (John.device_protection and device_protection.disabled) |
| 3   | E<> (John.screen_locking and screen_locking.turned_on) |
| 4   | E<> (John.screen_locking and screen_locking.turned_off) |
| 5   | E<> (John.timeout and timeout.short) |
| 6   | E<> (John.timeout and timeout.long) |
| 7   | E<> (John.updating_mechanism and updating_mechanism关联交易) |
| 8   | E<> (John.updating_mechanism and updating_mechanism.关联交易) |
| 9   | E<> (John.time_to_reboot and time_to_reboot.right_away) |
| 10  | E<> (John.time_to_reboot and time_to_reboot.after_away) |
| 11  | E<> (John.password_age and password_age.non_expired) |
| 12  | E<> (John.password_age and password_age.expired) |
| 13  | E<> (John.password_length and password_length.long) |
| 14  | E<> (John.password_length and password_length.short) |
| 15  | E<> (John.password_reusability and password_reusability.unused) |
| 16  | E<> (John.password_reusability and password_reusability.used) |
| 17  | E<> (John.spot_signs and spot_signs.yes) |
| 18  | E<> (John.spot_signs and spot_signs.no) |
| 19  | E<> (John.report_threat and report_threat.yes) |
| 20  | E<> (John.report_threat and report_threat.no) |
| 21  | E<> (John.security_policy and security_policy.complied) |
| 22  | E<> (John.security_policy and security_policy.violated) |
| No. | Security Linear-Time Property                                                        |
|-----|--------------------------------------------------------------------------------------|
| 1   | E<> (Sara.device_protection and device_protection.disabled)                          |
| 2   | E<> (Sara.device_protection and device_protection.enabled)                           |
| 3   | E<> (Sara.screen_locking and screen_locking.turned_off)                              |
| 4   | E<> (Sara.screen_locking and screen_locking.turned_on)                               |
| 5   | E<> (Sara.updating_mechanism and updating_mechanism.manually)                        |
| 6   | E<> (Sara.updating_mechanism and updating_mechanism.automatically)                   |
| 7   | E<> (Sara.time_to_update and time_to_update.long)                                    |
| 8   | E<> (Sara.time_to_update and time_to_update.short)                                   |
| 9   | E<> (Sara.time_to_reboot and time_to_reboot.after awhile)                            |
| 10  | E<> (Sara.time_to_reboot and time_to_reboot.right_away)                              |
| 11  | E<> (Sara.password_age and password_age.expired)                                     |
| 12  | E<> (Sara.password_age and password_age.non_expired)                                 |
| 13  | E<> (Sara.password_length and password_length.short)                                 |
| 14  | E<> (Sara.password_length and password_length.long)                                  |
| 15  | E<> (Sara.password_reusability and password_reusability.used)                        |
| 16  | E<> (Sara.password_reusability and password_reusability.unused)                      |
| 17  | E<> (Sara.spot_signs and spot_signs.yes)                                             |
| 18  | E<> (Sara.spot_signs and spot_signs.no)                                              |
| 19  | E<> (Sara.report_threat and report_threat.no)                                        |
| 20  | E<> (Sara.report_threat and report_threat.yes)                                       |
| 21  | E<> (Sara.security_policy and security_policy.violated)                              |
| 22  | E<> (Sara.security_policy and security_policy.complied)                              |
Table 8.3: Knowledge base + Bella’s security properties

| No. | Security Linear-Time Property                                                                 |
|-----|----------------------------------------------------------------------------------------------|
| 1   | E<> (Bella.device_protection and device_protection.disabled)                                   |
| 2   | E<> (Bella.device_protection and device_protection.enabled)                                   |
| 3   | E<> (Bella.screen_locking and screen_locking.turned_on)                                       |
| 4   | E<> (Bella.screen_locking and screen_locking.turned_off)                                      |
| 5   | E<> (Bella.timeout and timeout.long)                                                           |
| 6   | E<> (Bella.timeout and timeout.short)                                                           |
| 7   | E<> (Bella.updating_mechanism and updating_mechanism.manually)                                |
| 8   | E<> (Bella.updating_mechanism and updating_mechanism.automatically)                            |
| 9   | E<> (Bella.time_to_update and time_to_update.short)                                             |
| 10  | E<> (Bella.time_to_update and time_to_update.long)                                              |
| 11  | E<> (Bella.time_to_reboot and time_to_reboot.right_away)                                       |
| 12  | E<> (Bella.time_to_reboot and time_to_reboot.after_away)                                       |
| 13  | E<> (Bella.password_age and password_age.expired)                                               |
| 14  | E<> (Bella.password_age and password_age.non_expired)                                           |
| 15  | E<> (Bella.password_length and password_length.long)                                            |
| 16  | E<> (Bella.password_length and password_length.short)                                           |
| 17  | E<> (Bella.password_reusability and password_reusability.unused)                               |
| 18  | E<> (Bella.password_reusability and password_reusability.used)                                 |
| 19  | E<> (Bella.spot_signs and spot_signs.yes)                                                       |
| 20  | E<> (Bella.spot_signs and spot_signs.no)                                                        |
| 21  | E<> (Bella.report_threat and report_threat.yes)                                                 |
| 22  | E<> (Bella.report_threat and report_threat.no)                                                  |
| 23  | E<> (Bella.security_policy and security_policy.complied)                                        |
| 24  | E<> (Bella.security_policy and security_policy.violated)                                        |

Table 8.5: Knowledge base + Zac’s security properties

| No. | Security Linear-Time Property                                                                 |
|-----|----------------------------------------------------------------------------------------------|
| 1   | E<> (Zac.device_protection and device_protection.enabled)                                     |
| 2   | E<> (Zac.device_protection and device_protection.disabled)                                    |
| 3   | E<> (Zac.screen_locking and screen_locking.turned_on)                                         |
| 4   | E<> (Zac.screen_locking and screen_locking.turned_off)                                        |
| 5   | E<> (Zac.timeout and timeout.long)                                                            |
| 6   | E<> (Zac.timeout and timeout.short)                                                            |
| 7   | E<> (Zac.spot_signs and spot_signs.no)                                                         |
| 8   | E<> (Zac.spot_signs and spot_signs.yes)                                                         |
| 9   | E<> (Zac.report_threat and report_threat.no)                                                   |
| 10  | E<> (Zac.report_threat and report_threat.yes)                                                  |
| 11  | E<> (Zac.security_policy and security_policy.complied)                                         |
| 12  | E<> (Zac.security_policy and security_policy.violated)                                         |
Table 8.1: John’s reachability analysis results

| No. | Property Status |
|-----|-----------------|
| 1   | satisfied       |
| 2   | not satisfied   |
| 3   | satisfied       |
| 4   | not satisfied   |
| 5   | satisfied       |
| 6   | not satisfied   |
| 7   | satisfied       |
| 8   | not satisfied   |
| 9   | satisfied       |
| 10  | not satisfied   |
| 11  | satisfied       |
| 12  | not satisfied   |
| 13  | satisfied       |
| 14  | not satisfied   |
| 15  | satisfied       |
| 16  | not satisfied   |
| 17  | satisfied       |
| 18  | not satisfied   |
| 19  | satisfied       |
| 20  | not satisfied   |
| 21  | satisfied       |
| 22  | not satisfied   |

Table 8.7: Knowledge base + Zoe’s security properties

| No. | Security Linear-Time Property |
|-----|--------------------------------|
| 1   | E<> (Zoe.updating_mechanism and updating_mechanism.automatically) |
| 2   | E<> (Zoe.updating_mechanism and updating_mechanism.manually)     |
| 3   | E<> (Zoe.time_to_reboot and time_to_reboot.after_while)          |
| 4   | E<> (Zoe.time_to_reboot and time_to_reboot.right_away)           |
| 5   | E<> (Zoe.password_age and password_age.non_expired)              |
| 6   | E<> (Zoe.password_age and password_age.expired)                  |
| 7   | E<> (Zoe.password_length and password_length.short)              |
| 8   | E<> (Zoe.password_length and password_length.long)               |
| 9   | E<> (Zoe.password_reusability and password_reusability.unused)   |
| 10  | E<> (Zoe.password_reusability and password_reusability.used)     |
Table 8.2: Sara’s reachability analysis results

| No. | Property Status |
|-----|----------------|
| 1   | satisfied       |
| 2   | not satisfied   |
| 3   | satisfied       |
| 4   | not satisfied   |
| 5   | satisfied       |
| 6   | not satisfied   |
| 7   | satisfied       |
| 8   | not satisfied   |
| 9   | satisfied       |
| 10  | not satisfied   |
| 11  | satisfied       |
| 12  | not satisfied   |
| 13  | satisfied       |
| 14  | not satisfied   |
| 15  | satisfied       |
| 16  | not satisfied   |
| 17  | satisfied       |
| 18  | not satisfied   |
| 19  | satisfied       |
| 20  | not satisfied   |
| 21  | satisfied       |
| 22  | not satisfied   |
Table 8.3: Bella’s reachability analysis results

| No. | Property Status |
|-----|-----------------|
| 1   | satisfied       |
| 2   | not satisfied   |
| 3   | satisfied       |
| 4   | not satisfied   |
| 5   | satisfied       |
| 6   | not satisfied   |
| 7   | satisfied       |
| 8   | not satisfied   |
| 9   | satisfied       |
| 10  | not satisfied   |
| 11  | satisfied       |
| 12  | not satisfied   |
| 13  | satisfied       |
| 14  | not satisfied   |
| 15  | not satisfied   |
| 16  | not satisfied   |
| 17  | not satisfied   |
| 18  | not satisfied   |
| 19  | satisfied       |
| 20  | not satisfied   |
| 21  | satisfied       |
| 22  | not satisfied   |
| 23  | satisfied       |
| 24  | not satisfied   |
Table 8.4: Zac’s reachability analysis results

| No. | Property Status     |
|-----|---------------------|
| 1   | satisfied           |
| 2   | not satisfied       |
| 3   | satisfied           |
| 4   | not satisfied       |
| 5   | satisfied           |
| 6   | not satisfied       |
| 7   | satisfied           |
| 8   | not satisfied       |
| 9   | satisfied           |
| 10  | not satisfied       |
| 11  | satisfied           |
| 12  | not satisfied       |

Table 8.5: Zoe’s reachability analysis results

| No. | Property Status     |
|-----|---------------------|
| 1   | satisfied           |
| 2   | not satisfied       |
| 3   | satisfied           |
| 4   | not satisfied       |
| 5   | satisfied           |
| 6   | not satisfied       |
| 7   | satisfied           |
| 8   | not satisfied       |
| 9   | satisfied           |
| 10  | not satisfied       |