5-2023

Modeling User Behavior for Cyber Security with Formal Methods and Agent based Simulation

Hamdah Albalawi

Modeling User Behavior for Cyber Security with Formal Methods and Agent based Simulation

by

Hamdah Albalawi

Bachelor of Science
in Computer
king khalid university
2015

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

Master of Science
in
Computer Science

Melbourne, Florida
May, 2023

We the undersigned committee
hereby approve the attached thesis

Modeling User Behavior for Cyber Security with Formal Methods and Agent based Simulation by Hamdah Albalawi

---

Siddhartha Bhattacharyya, Ph.D.
Associate Professor
Computer Engineering and Sciences
Major Advisor

Marius Silaghi, Ph.D.
Professor
Computer Engineering and Sciences

Juan Camilo Avendano, Ph.D.
Graduate Faculty
Computer Engineering and Sciences

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

ABSTRACT

Title:

Modeling User Behavior for Cyber Security with Formal Methods and Agent based Simulation

Author:

Hamdah Albalawi

Major Advisor:

Siddhartha Bhattacharyya, Ph.D.

Despite society's positive outlook, technology poses real cyber security threats. Technology's benefits can sometimes make it difficult to believe that potential threats lurk behind every device and platform. As cybercrime rises, we have come to rely increasingly on flawed devices and services. As cyberattacks become more prevalent, security professionals are committed to developing more robust and dependable security solutions. In cyber security, human error is regarded as the weakest link since all technical security solutions are vulnerable to human error. Among other human characteristics, risk-taking, logical decision-making, extraversion, and gender can significantly affect cyber security. However, there still is the need to conduct research on improving security based on user behavior, situational awareness and then accordingly assigning security policies. Our contribution, in this research effort has been in the development of a framework to support the generation and implementation of cybersecurity policies specific to the needs of different users or based on users cybersecurity behavior. In the process, we were able to identify the crucial characteristics of user security behavior. Then, using Formal Methods based environment, Uppaal, we modeled the specified security behaviors. Next, we mapped user security behavior to NIST assurance levels. In doing so, we expanded the previous research outcomes, by adding the NIST assurance

levels to include issue-specific policy assurance, proactive awareness, social media, and website access. Finally, the formal models were mapped to an agent based simulation environment using NetLogo. Policies were generated based on end user security behaviors. Therefore, we were able to determine types of security policies an organization or other entity should impose on specific users.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

FSA	Finite-State Automata
SeBIS	Security Behavior Intentions Scale
USA	United State of America
DoSpeRT	Domain Specific Risk Attitude
USPGF	User-Specific Policy Generation Framework
TA	Timed Automaton
TCTL	Timed Computation Tree Logic
CTL	Computation Tree Logic
DS	Device Securement
PG	Password Generation
PA	Proactive Awareness
U	Updating
OTP	One Time Password
NIST	National Institute of Standards and Technology
PEP	policy enforcement point
PDP	policy decision point
PE	policy engine
XACML	Access Control Markup Language

Acknowledgements

First, I want to express my sincere gratitude to my beloved country, the Kingdom of Saudi Arabia, for giving me this opportunity to pursue my education and realize one of my greatest dreams.

Second, I am extremely grateful to my supervisor, Dr. Siddhartha Bhattacharyya, for giving me the opportunity to work on this thesis through his invaluable guidance and advice. The multitude of methods he has used to assist, inspire, and morally support me have been instrumental in my ability to complete this thesis.

Finally, I would like to thank the members of my committee, Juan C. Avendano Arbelaez and Marius Silaghi. For their encouragement, valuable advice, and for taking the time out of their busy schedules to serve on my thesis advisory committee, I am grateful to my committee members.

Dedication

This thesis is dedicated to my parents, Awad and Mokidah Albalawi. Thank you so much! I cannot express my appreciation and gratitude enough to you. The support and inspiration you have provided have been invaluable. Throughout my life, you have taught me to be unique, and determined, to believe in myself, and to persist through any challenge. I appreciate the unconditional love and support you have shown me throughout this journey. I will always and forever love you.

Chapter 1

Introduction

Today's world is increasingly dependent on technology. Among the benefits of this trend is instant information access on the Internet and innovative home automation technology and concepts like the Internet of Things. The good that technology brings can make it hard to believe that potential threats are stalking behind every device and platform. Modern technology presents real cyber security threats despite society's positive perceptions. In a world where cybercrime is on the rise, we have come to rely increasingly on devices and services that have flaws. To be more prepared for cyberattacks, security professionals remain committed to developing more robust and reliable security solutions. *Zero Trust* is a 2009 Forester alumnus John Kindervag's idea that centered on the premise that trust is a weakness, and that security must be created with the approach "Never trust, always verify." [19]. *Zero Trust* is a 2009 Forester alumnus John Kindervag's idea that centered on the premise that trust is a weakness, and that security must be created with the approach "Never trust, always verify." [19] In *Zero Trust*approaches, no network participant is trusted, and any access to organizational resources can pose a threat. As a result, every single access is inspected and verified. An access request is

only granted after it has been verified. The user can be granted access to a full range of functions or data or only those for which he or she is authorized. It is not sufficient to verify only using the password but also considering other factors and information sources: the user's password, the device used, the location and time (Radwan (Gratian et al.,2020) [26], and access privileges (Yan Wang, 2020) [38]. Organizations, in general, have no default trust in any user or device, whether inside or beyond the network's perimeter. This guarantees that everyone is subject to stringent access controls, constant authentication, and authorization, and that all actions they conduct are allowed. For example, if an attacker gains prolonged network access in order to obtain valuable access to secret data, any lateral movement within the network is more likely to be noticed and rejected by the internal security access control policy [19] .

Human error is considered the weakest link in cyber security since all technical security solutions are subject to failure due to human error. Human qualities such as financial risk-taking, logical decision-making, extraversion, and gender, among others, can have a significant impact on cyber security. A survey conducted by (Gratian et al., 2018) of 369 students, faculty, and staff at a large public university found that cyber security behavior intentions vary by 5 percent to 23 percent [16]. Cybersecurity research has mainly focused on improving computer network systems, since many believe that information technology advancements and software development are the primary means to increase security. A limited amount of research has been conducted on improving system analysts' cognitive abilities and situational awareness. A cyber attacker, however, uses social engineering methods to break into a computer system or network, rather than a computer system itself. These methods include social engineering (e.g., tricking users into gathering information) and cognitive hacking (e.g., spreading misinformation). [4]

.

Occasionally, a breach caused by human error, or a system breakdown costs less than one caused by a cybercriminal or a malevolent insider, but human irresponsibility should not be minimized. In a study conducted by the Ponemon Institute [31], 507 companies in 16 countries were interviewed regarding data breaches that occurred between July 2018 and April 2019. Approximately 24 percent of data breaches are caused by user irresponsibility, which results in an average financial loss of 3.5 million. Inadvertent insiders can cause unintentional and unintentional breaches due to phishing or infection, lost, or stolen equipment. According to the Ponemon Institute, human error contributes to a data breach costing enterprises 133 on average per compromised record. Detecting and containing such a breach takes enterprises approximately 181 days and 61 days, respectively [31].

Researchers have found that end-user security decisions are influenced by demographics, personality factors, decision-making styles, and risk-taking attitudes. (Al-Qadheeb et al., 2021) provide a method for strengthening user control in a zero-trust environment by evaluating security behavior and developing user-specific policies. A formal method-based framework is created to assist in defining user-specific security policies. Starting with Identifying and selecting critical features that define how people behave in terms of security. They then formalized the security behaviors they discovered to perform automated reasoning. Finally, they were able to identify security flaws in how people behave and then propose solutions for addressing them. [5] "Enhancing Cybersecurity by Generating User-Specific Security Policies through the Formal Modeling of User Behavior". As a part of their study, (AlQadheeb et al., 2021) analyzed four users' behaviors: password generation, proactive awareness, updating devices, and device security. In this paper, we discuss social media suspicious websites and identity as security behaviors.

In our research, we explore the question of how to automatically establish trustworthy and accurate security policies. We do this by examining and evaluating security behaviors displayed by users in a *Zero Trust* environment.

Q1: What are the possible extensions to end-user security behaviors?

Q2: How can we map the end user behavior to NIST assurance levels? Additionally, addressing the complexities when modeling and simulation environment is different?

Q3: How to implement security policies in an agent-based environment?

As part of our contribution, we have developed a methodology-based framework to help develop user-specific cybersecurity policies. By doing so, we were able to identify the crucial characteristics that characterize the security behavior of users. Using Uppaal as a formal model, we modeled the specified security behaviors. Next, we mapped user security behavior to NIST assurance levels. We expanded the NIST assurance level model to include issue-specific policy assurance levels mapped to proactive awareness, social media, and website access. Our last step was to simulate the security behaviors of end users in NetLogo to generate policies.

This thesis is organized as follows: Chapter 2 provides background information and a discussion of related works. Chapter 3 describes our research methodology and what we did to achieve our goal. Chapter 4 discusses extending and modeling user security predictors and linear time security properties. Chapter 5 discuss how to generate security policies that are specific to each user using Netlogo. Chapter 6 outlines the conclusions and future work.

Chapter 2

Literature Survey

A feeling of security is based more on your psychological reaction to threats and countermeasures than on probability or scientific calculations. To design security solutions, it is essential to understand how our brains work and how they fail. [35]

In the design of security systems, human elements should be incorporated because hackers pay more attention to the human link in the chain than security designers. [3] Parker emphasizes the need-to-know principle as a precept of password security, and that users' insecure conduct stems from a lack of knowledge about password methods, content, and cracking. Incompatible password systems may lead users to try to get around them, lowering overall security. Consider using a shared password to restrict access to shared information. Security researchers have identified insecure user behavior and low-security motivation as key issues that must be addressed [3] [29] .

According to (West et al., 2009), Users' engagement with the security system and security decisions are influenced by human and cognitive factors. The authors claim that humans are exploitable, and there is plenty of evidence to support this claim. The IT Policy Compliance Group states that the majority of organizations that have experienced

data abuse or theft consider their internal staff to be at fault, according to their research. It is rare for them to tie malicious behaviors, such as hacking, attacks, and viruses, to a specific cause.The end-users are responsible for almost all security-related problems due to slower technological adoption, a misunderstanding of risk and its effect on their actions, and inadequate security decision-making. [35]

Organizations today must maintain a delicate balance between adopting new technologies and protecting their systems. The fundamental security idea of ”never trust, always verify” was significantly emphasized in 2010 by John Kindervag, a Security and Risk Principal Analyst at Forrester Research Inc. The human behaviors will also need to be taken into consideration by organizations thinking about Zero Trust policy. Humans' ability to digest information is limited, their natural psychology is based on past experiences, and they tend to rely on those experiences when making security decisions.

In [5], (AlQadheeb et al., 2021) provide an approach for increasing user control by examining security behavior and creating user-specific policies in a zero-trust environment. Their main contribution is developing a formal method-based framework to help with the process of making user-specific security policies. In the process, they showed how to find and choose the key characteristics that define how users behave in terms of security. Then, they turned the security behaviors they had found into formal models so that automated reasoning could be done. Lastly, they were able to find security flaws in how users act and then suggest policies to fix them. The scale developed by the researchers focus on four constructs: password generation, proactive awareness, updating, and setting up a lock screen using patterns, passwords, or PINs as part of device securement.

In an effort to build on (AlQadheeb et al., 2021) work, (Schechinger,2021) conducted Additional study to see if agent-based modeling can be expanded to incorporate

enhanced user actions to further simulate user behavior in a Zero Trust environment. Where subjects in an untrusted zone request access to resources protected by a policy enforcement point in a trust zone. Access is granted or denied based on a trust algorithm. As soon as the model gathers (produces) all of the assurance levels, a trust algorithm determines if the generated scores are permitted to access the resource. The assurance levels are roughly based on the assurance levels in NIST standards. Modeling with NetLogo demonstrates that this tool it can be used to model and simulate zero trust architecture and associated user behaviors. [13]

(Moustafa et al., 2021) review current research on psychological traits and individual variations in computer system users that explain their vulnerability to cybersecurity threats. The study identifies research gaps and propose psychological strategies to assist computer system users in complying with security regulations, in turn improving network and information security. According to (Moustafa et al., 2021), most cybersecurity research has focused on enhancing computer networks, primarily because information technology advancements and software development are expected to increase information security. In contrast, few studies have attempted to improve cognitive abilities and situational awareness of system analysts. A number of human errors related to cyber and network security are common, such as sharing passwords, oversharing information on social media, accessing suspect websites, using unauthorized external media, reusing the same password across multiple sites, opening attachments from untrustworthy sources, sending sensitive information over mobile networks, failing to physically secure personal electronic devices, and not updating software. [4]

Agent-based simulation (ABS) is a simulation tool that simulates systems using autonomous, interacting ”agents.” It has many applications, including supply chain and stock market modeling, anticipating healthcare future demands, and social psychology.

Agent-based simulation (ABS) is a simulation tool that simulates systems using autonomous, interacting ”agents.” It has many applications, including supply chain and stock market modeling, anticipating healthcare future demands, and social psychology. It will be easier to understand and solve problems when the problem can be modeled as being composed of agents, when the agents' behavior is well defined, when they can adapt and change their behavior, when they have the ability to learn and engage in dynamic strategic interactions, and when the agents have the ability to change their relationships. In agricultural systems, agent-based simulation models can be used to improve social psychology theory development and examine how humans interact with their environments. [21]

2.1 Background

Our expectations for the formal model are initially expressed in natural language, which is then transformed into a more formal and well-defined language that computers can interpret and refine [31]. First, we'll discuss the fundamental concept of linear-time properties, as well as a few significant, albeit relatively simple, types of these properties. Secondly, we describe how Uppaal encodes model specifications in formal language. Uppaal employs a formal language known as Timed Computation Tree Logic (TCTL). We conclude by evaluating whether our developed model is valid based on the security properties of our knowledge base generated by Uppaal.

2.1.1 Overview of Linear-Time Properties

A linear-time property is an essential prerequisite for the traces of transition systems [8]. 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 ω-language, set of infinite-length sequences of symbols, over 2^{AP} where 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 [8]. 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 AP and T be a transition system over AP too in a system without terminal states, then in this case T satisfies P, expressed as:

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

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

$$\pi \models P, \text{ when } Traces(\pi) \in P.$$

3. Letting s be a state of T, it translates that s satisfies P, expressed as:

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

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 $T1$ and $T2$ have a similar set of traces, then it is expected that they satisfy the same Linear-time properties. Assuming $T1$ and $T2$ have similar traces and let P be an arbitrary linear-time property, if in any case $T1 \models P$, then $Traces(T1) \subseteq P$, implying that $Traces(T2) \subseteq P$ since $Traces(T1) = Traces(T2)$. When dealing with software design, it is essential to note the above expressions because if $Traces(T1) \subseteq Traces(T2)$, then $T1$ is a correct implementation of $T2$. $T2$ is considered an abstract model, while $T1$ is treated as one particular implementation. Generalizing the above statement, $T1$ cannot have properties that are not exhibited in $T2$.

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 [8]. 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:

$$Trace(\pi) = L(s_0)\ L(s_1)\ \ldots.$$

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 α. 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.

2.1.2 Invariant

An invariant refers to a linear-time property that is provided by Φ condition for any given states and remains true for all the reachable states. A good example is described by in mutex property $\Phi = \neg\ crit_1 \vee \neg\ crit_2$. 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 Φ *AP*, then a linear-time property P_{inv} over *AP* is an invariant in the sense that: $P_{inv} = \{A0\ A1\ A2\ldots\ \in (2^{AP})^{\omega} \mid \forall j \geq 0.Aj \models \Phi\}$, where Φ is referred to as the invariant condition of P_{innv}.

2.1.3 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 [8]. 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 $\Phi * (\neg \Phi)$. 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 Φ [36]. 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 [36]. 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.

2.2 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) [**?**]. The syntax of the Timed Computation Tree Logic is expressed as follows:

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

- 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 2.1 from [33], we can see the different combinations of path formulae and state formulae accepted by Uppaal. Figures 2.1 to 2.4 represent the computation trees of the formulae.

Table 2.1: Temporal connectives in Uppaal

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

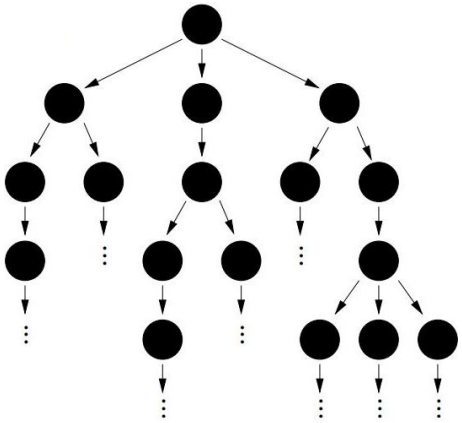


Figure 2.1: AG computation tree

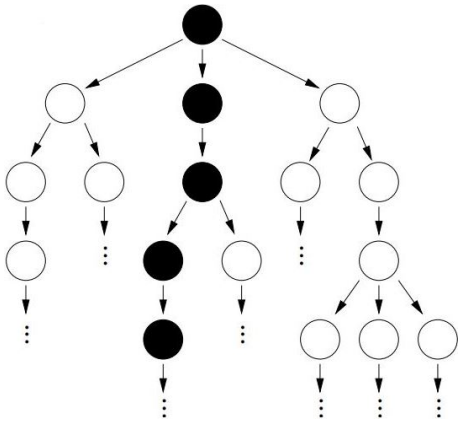


Figure 2.2: EG computation tree

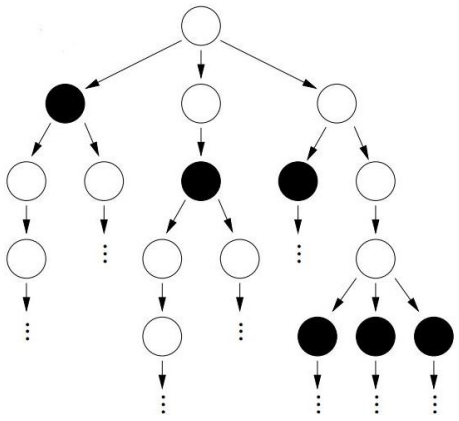


Figure 2.3: AF computation tree

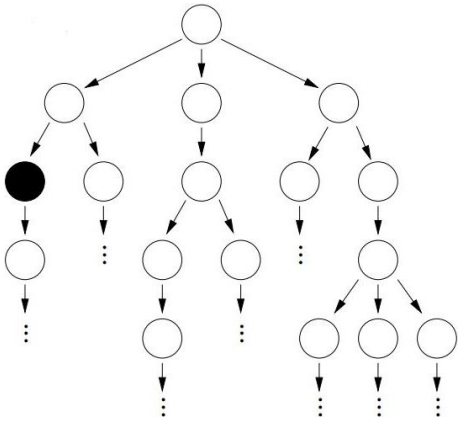


Figure 2.4: EF computation tree

2.3 Knowledge Base Properties

We decided to define unique properties for each component of the knowledge base after thoroughly understanding Uppaal's definition of properties as Timed CTL formulas with only four temporal operators i.e., A, E, G, F). We will provide additional examples in sections 4.6.1 to 4.6.4 explaining why and how we selected the most critical issues within various security settings such as device securement, password generation, proactive awareness, updates, social media suspicious websites, and identity. We based our example creation on existing literature to determine the difference between good and bad decisions in various settings. In addition to our examples of good and poor security decisions, organizations can also implement their own policies to replace ours. Our ability to examine user behavior and identify poor security practices can be achieved using such properties. Policymakers can use our method to set their own policy thresholds for "bad" decisions.

Chapter 3

Research Methodology

This chapter discusses framework that can be used to implement user security polices based on zero trust. One of the initial steps involve identifying how to construct and verify a formal model based on users' security behaviors; this can the be used to determine what kind of security policy each individual will receive automatically. It can be difficult to select the best formal model development strategy due to the variety of techniques available, ranging in complexity from simple to quite complicated with several modules.

By developing the strategy, we aim to address some common problems with Internet users' poor security practices while, maintaining a high level of security. For this reason, we decided to set up our own strategy to ensure that every step of the process is implemented exactly. This will enable us to have proper checks to validate that the approach meets the objectives and provides the proper control needed to generate the cybersecurity policies from the outset. Prior to using our simulation approach and developing the required formal model, we created a framework that helped us remain focused and organized.

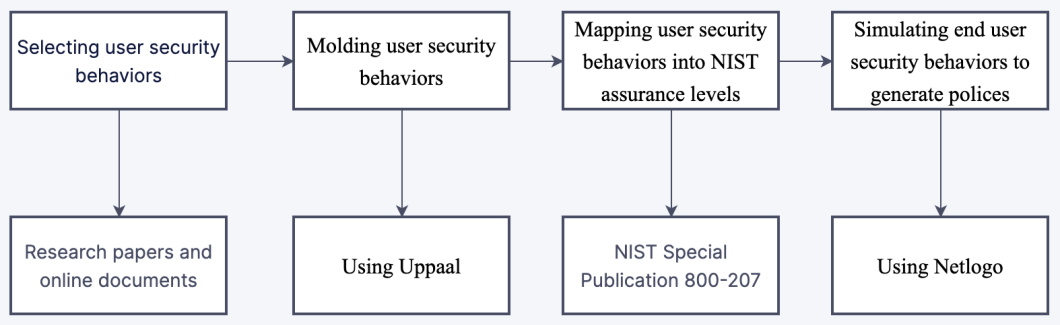


Figure 3.1: User-specific policy generation framework

This research is structured according to our User-specific policy generation framework (USGPF), shown in Figure 1. As shown in the figure, the development process is depicted as a four-stage chain. In each section, we provide more information about the tools and strategies we used to construct and validate the formal model and agent-based simulation. A detailed explanation of each step of the USPGF is presented in chapters 4 and 5.

The research methodology involves constructing a formal model of users' security behaviors, correlating those behaviors with NIST assurance levels, and then determining what type of policy should be automatically applied to each user. "Enhancing Cybersecurity by Generating User-Specific security Policies through the Formal Modeling of User Behavior" (AlQadheeb et al.,2021) examines four user behaviors: password generation, proactive awareness, updates, and device security. This paper discusses social media, suspicious website access, and identity as security behaviors.

Our essential contribution is the development of a formal method-based framework to aid in the process of developing user-specific cybersecurity policies. While doing so, we identified the crucial characteristics that characterize users' security behavior. Then, we modeled the specified security behaviors using Uppaal as a formal model.

Next, we mapped user security behavior into NIST assurance levels. We extended the NIST assurance level model to include an issue-specific policy assurance level mapped into proactive awareness, social media, and accessing suspicious websites. Finally, we simulated end-user security behaviors to generate policies using NetLogo.

Chapter 4

Extending and modeling user security predictor and Linear time security properties

4.1 Extending user security Predictor

In this section, we formalize the user's security-related behavior by considering the user's security decisions. It explains the extent of data collection, comparison, and selection to create a reliable knowledge base that meets the requirements for building a formal model, which was achieved after considering various options.

Previous research have thoroughly studied the relationships between individual differences in demographics, personality traits, decision-making styles, and risk-taking preferences and their influence on users' security-related actions in cyberspace, as illustrated in Chapter 2. We focused on decision-making since it has been demonstrated

in experiments that it predicts security practices more accurately than demographics, personality factors, and risk-taking.

Choosing how to react to a particular event requires assessing risks, considering potential consequences, and exploring alternatives. A person's ability to make appropriate decisions is limited by the fundamental structure of their psychology, their limited ability to process information, and their almost total reliance on past experience. [35]

It is difficult to distinguish between what is appropriate and what is not in cybersecurity. It is more likely that a crafty individual will target individuals leaving their devices unattended in public, creating passwords that are easy to guess, visiting suspicious websites in a rush, or forgetting to identify themselves. (AlQadheeb et al.,2021) examines four user behaviors: password generation, proactive awareness, updates, and device security. This paper extends the previous approach by adding social media, suspicious websites, and identity as security behaviors.

4.2 Modeling User Security Behavior

We analyzed studies on how decision-making processes related to security behavior in section 4.1 to establish the proper knowledge base and identify the most relevant requirements for describing how users interact differently with security systems. We will review the practical factors we considered before implementing the formal model in the first section of this section. Secondly, we outline the method for converting requirements into formal models. As a result of this study, we believe that graph-based knowledge representations of Finite-State Automata (FSA) are the most effective way to model user security behavior.

4.2.1 Model Considerations

Modeling human-machine interactions is challenging due to the complexity and uncertainty of human behavior. So we choose a subset of possible behaviors from a wide variety of knowledge requirements. In order to gather the requirements for modeling, we examined the aspects of a user's security-related behavior that is relevant to device security, social media suspicious websites, and identity, which is challenging. In order to simulate user behavior across the many parts of SeBIS (Security Behavior Intentions Scale), we built an architecture that allows decomposition of behavior. This is shown in figure 4.1, which is a graphical representation of the knowledge base architecture, it shows how the model is structured and organized.

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, we checked that each aspect of the model is covered, and ensured 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, a deeper investigation is needed to identify the interactions between the behaviors.

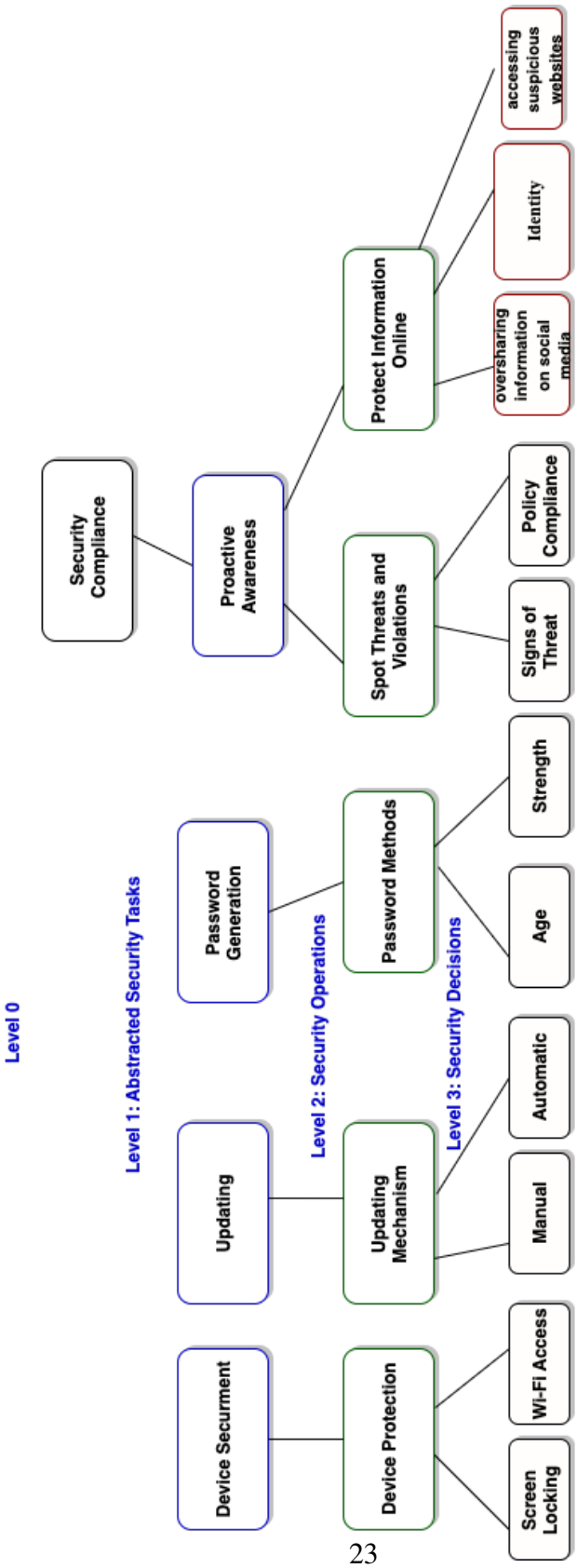


Figure 4.1: Knowledge base architecture

4.2.2 Levels of Abstraction

In this section, we provide detailed explanations of each level of abstraction.

- **Layer (0)** is the most abstract of the 5 levels of security service check. It incorporates SeBIS concepts such as security compliance.
- **Layer (1)** is the second most abstract of the four security service verification layers. It simulates the SeBIS concepts outlined by Egelman and Peer: device security, password generation, proactive awareness, and updating.
- **Layer (2)** divides SeBIS concepts into five security settings (device protection, update method, password methods, spot threat and violation, and protect information online), depending on user preference, whether to check for additional sub-services.
- **Layer (3)** has detailed information based on what the user chooses. This level further adds on the behaviors to enable device protection, update method, password methods, spot threat and violation, and protect information online .

4.3 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) [28].

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 [12], Uppaal [33], PVS [27], and Z3 [24] were considered carefully. We chose Uppaal [10][20][33], 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.

4.3.1 Mathematical Representation Within the Model Paradigm.

Uppaal uses timed automata [6], 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 [11].

- **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 $\langle S, s_0, \rightarrow \rangle$, where $S \subseteq L \times \mathbb{R}^C$ is the set of states, $s_0 = (l_0, u_0)$ is the initial state, and $\rightarrow \subseteq S \times \{\mathbb{R}_{\geq 0} \cup A\} \times S$ 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 $exec(TA)$ is the sequence of consecutive transitions, while the set of execution traces of the TA is denoted by $traces(TA)$.

4.3.2 Password Generation

- **OTP**

  Known as an OTP, one-time passwords are passwords that are valid for only one time. With one-time passwords, security can be improved, and strong authentication can be supported. Furthermore, 47 percent of people use a password that is more than 5 years old, and 17 percent of people use '123456' as their password. Even worse, 95 percent of people share no fewer than six passwords with their friends [18]. Even though your users use strong passwords and never write them down, they can still be compromised by keylogging malware or man-in-the-middle attacks. So, how do you help protect your most valuable assets against poor password hygiene or electronic eavesdropping? One way to do this is through the use of a 'one-time password,' which is a disposable password valid only once.

  Table 4.1 lists the equivalent expressions of OTP in Uppaal specification language. In Figure 4.2, we can notice the state-transition graphs of password generation aspects.

Table 4.1: Password generation properties

No.	Knowledge Base Property
1	`E<> (OTD.no)`
2	`E<> (OTD.yes)`

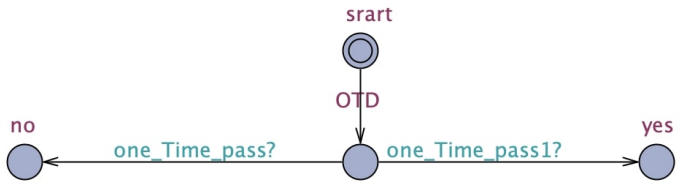


Figure 4.2: Password generation state-transition graphs

4.3.3 Identity

- **Identity Provided**

  Identity Provided is the lowest level of identity where the user provides any form of identity. What is your identity on the Internet? Your identity is formed by your personal characteristics, such as where you were born, when you were born, where you attended school, what size shoes you wear, etc. While some of those characteristics remain constant, such as your birthday, others, such as your hair color, change over time. A person's online identity is the sum of their characteristics and interactions on the Internet. Depending on how you interact with each website, each site will have a different view of who you are and what you do. Various representations of you are sometimes called partial identities owing to the fact that none of them accurately depicts who you are. As each website views you and your characteristics differently, each has a different idea of who you are.

- **Identity Self Asserted**

  Self-Attested provides the lowest level of assurance because it does not require a confirmation from authoritative sources. Individuals use this method to self-certify that they are who they say they are by photocopying their ID document, signing it, and writing "true copy" or "self-attested." By today's standards, this

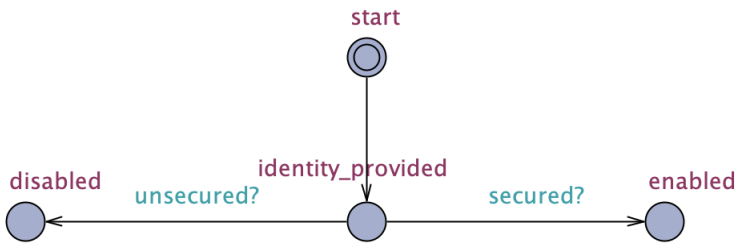
verification method is not considered adequate. Nonetheless, many businesses use this method, particularly those where employees do not interact directly with customers.[39]

- **Identity Based Knowledge**

Identity Based Knowledge requires users to provide basic information such as their name, address, or phone number before conducting a transaction or accessing other data. Following that, the information provided is compared with what is commonly considered ”knowledge” in various public databases. If a potential customer’s information does not match what is kept in the database, they are flagged as a security threat [32].

Table 4.2: Identity properties

No.	Knowledge Base Property
1	`E<> (identity_provided.enabled)`
2	`E<> (identity_provided.disabled)`
3	`E<> (identity_self_asserted.enabled)`
4	`E<> (identity_self_asserted.disabled)`
5	`E<> (identity_based_knowledge.yes)`
6	`E<> (identity_based_knowledge.no)`



4.3.4 Social Media

- **Posts Per Hour Check**

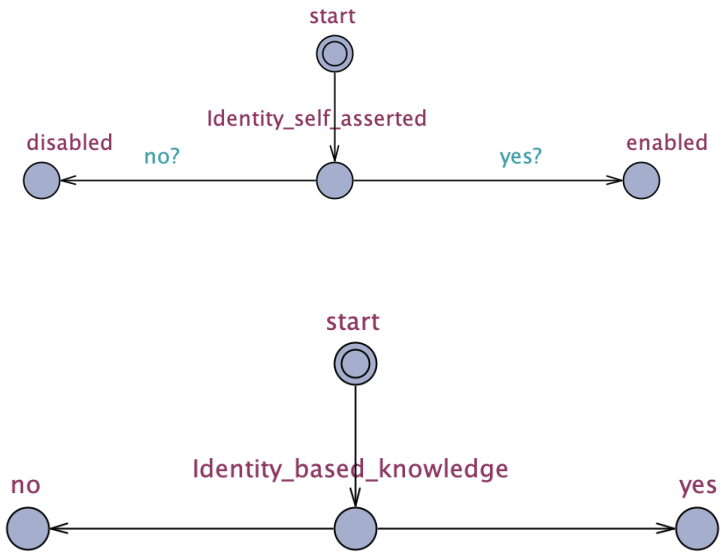


Figure 4.3: Identity state-transition graphs

Modern life is always surrounded by social media. The sharing of life experiences with loved ones can be enjoyable. The danger, however, is that if we're not careful, we might end up exposing more information than we intended or to a much broader audience than intended. [23] Posting every hour a post is considered oversharing in this setting. Criminals may be able to find out crucial information about you more easily if you post your personal information online. For instance: an image from a trip you are currently taking could alert criminals that you're away.

- **Sending Sensitive Information**

  While it can be tempting, some things are better kept private on social media and message boards. If certain pieces of personally identifiable information fall into the wrong hands, you could suffer identity theft, phishing, or other forms of fraud.

- **Open/Closed Social Network**

Social networks and social media have two distinct sharing models. A closed model (shared with only a small group of people) and an open model (shared with everyone). Twitter runs on an open, shared-with-everyone model. The main purpose of this system is to disseminate information as widely as possible. Because the message or content is made available to anyone who wishes to receive it, it can be widely disseminated in a short period of time. Closed networks, such as Facebook and LinkedIn, require participants to explicitly give permission for others to see their messages and content. An implicit level of trust is built into the system. As a result, sharing on a closed social network can be more secure.

Table 4.3: Social Media properties

No.	Knowledge Base Property
1	E<> (Posts_Per_Hour_Check.over_sharing)
2	E<> (Posts_Per_Hour_Check.not_oversharing)
3	E<> (SendSen_info1.yes)
4	E<> (SendSen_info1.no)
5	E<> (open_closed_SocialNetwork.yes)
6	E<> (open_closed_SocialNetwork.no)

4.3.5 Accessing Suspicious Websites

- **Click on Links in Emails**

  Malicious websites, as the name implies, are intended to harm users. These are not eCommerce stores, financial institutions, or web applications that provide critical services securely. Their goal instead is to steal valuable data or infect visitor devices with malicious software. It is risky to click a link in an email, text message, or on a website. Unsolicited emails and text messages with links should

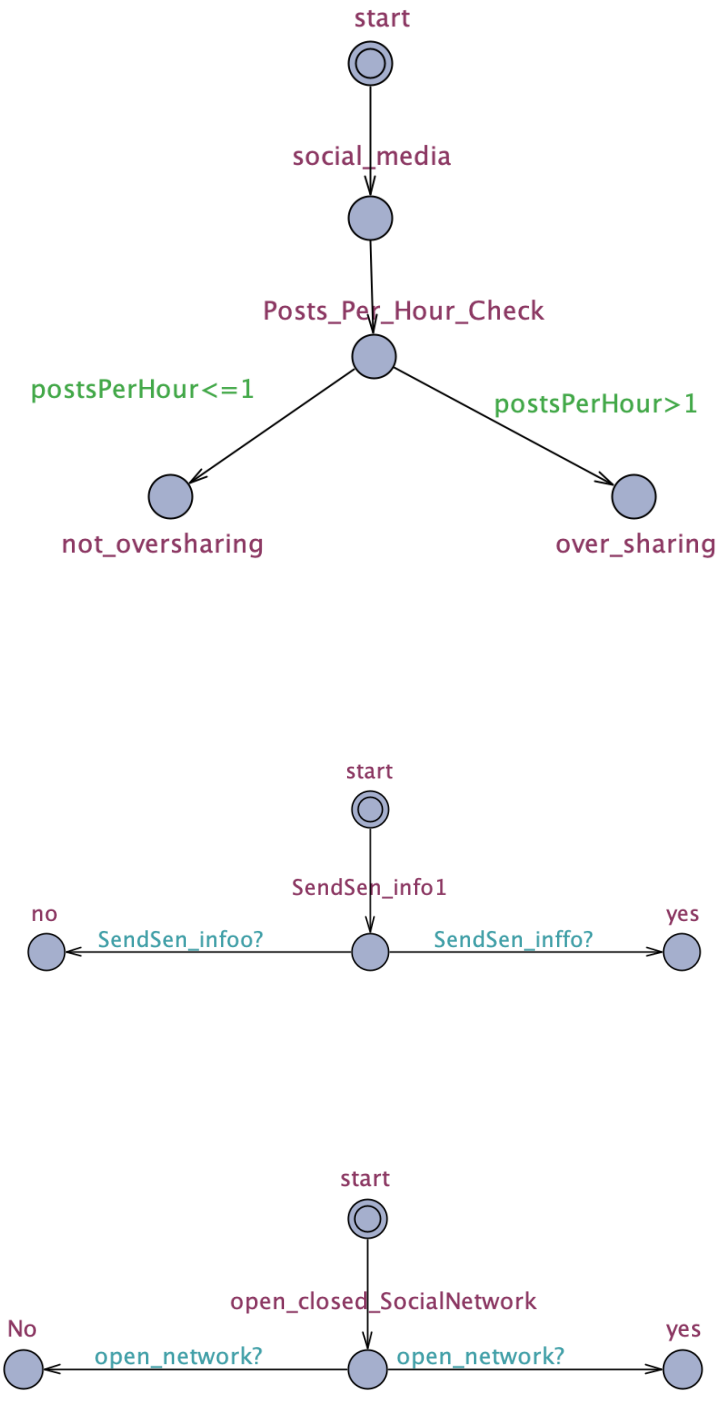


Figure 4.4: Social Media state-transition graphs

not be clicked. It is particularly common for hackers to insert links into emails that appear to come from legitimate companies. Avoid clicking on links if you are uncertain about their validity. You may end up in a place that raises red flags, or it may ask for information you don't need to provide [25]. We modeled some of the behaviors exhibited by users while accessing malicious websites as shown in Figure 4.5 .

- **URL Start with HTTPS**

  HTTPS ensures the security of your website. Your website's communication with your users' browsers is protected by HTTPS, which helps prevent intruders from interfering with communications between your website and theirs. Intruders can be malicious attackers or legitimate businesses that inject advertisements into your webpages. Many people believe that HTTPS is only necessary for websites that handle sensitive communications. Every unprotected HTTP request may reveal information about your users' behavior and identity. While one visit to an unprotected site may seem harmless, some intruders use aggregate browsing activities to deanonymize your users and infer their behaviors and intentions [9].

- **Use Anti Malware**

It's software that you knowingly install on your computer to prevent malware from infiltrating and infecting it. Anti-malware programs can accomplish this in three ways: by detecting malware on your computer, removing it, and repairing any damage it causes. As well, some Anti-Malware Premium products offer malicious website blocking and real-time protection, which means that the programs block websites designed to deliver malware as well as those that might be compromised by malware. Likewise, anti-malware runs continuously in the background to prevent malware from installing

on your system.

Table 4.4: Accessing Suspicious Websites properties

No.	Knowledge Base Property
1	E<> (click_links_in_emai.suspicious_websites)
2	E<> (click_links_in_emails1_normal_websites)
3	E<> (URL_start_with_https1.normal_websites
4	E<> (URL_start_with_http.s_suspicious_website)
5	E<> (use_anti_malware.s_suspicious_websites)
6	E<> (use_anti_malware.normal_websites)

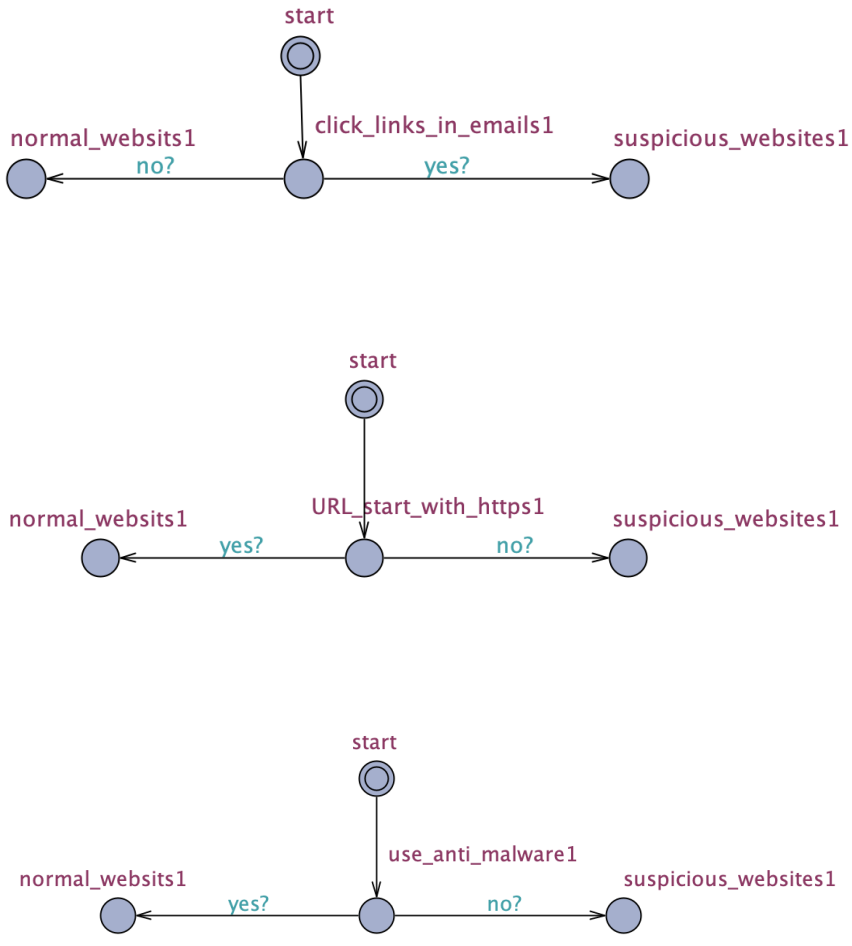


Figure 4.5: Malicious websites access state-transition graphs

Chapter 5

Generating User-Specific Security Policy Netlogo

To reduce the risks associated with cyberattacks, organizations have upgraded their tools and strategies in response to the current proliferation of cyberattacks. The zero trust architecture is a model and set of design guidelines that accepts the existence of threats both from inside and outside the network and assumes resource compromise. It is important to focus on user interactions when using this strategy because, in most environments, users are the weakest link. In this context, AlQadheeb et al.'s "Enhancing Cybersecurity by Generating User-Specific Security Policy through the Formal Modeling of User Behavior" (2021) describes a method for analyzing user security behavior and creating user-specific policies. Modeling end-user behavior with Finite-State Automation (FSA), creation of linear time security properties, and application of user-specific policies are all included.

To further develop this work, we conducted additional research to determine whether agent-based modeling can leverage enhanced user activities to model user behavior

within a Zero Trust environment. As part of the study, agent-based modeling was used to better understand user patterns and interactions. Agent-based modeling simulates social systems, such as user interactions, which are ”composed of agents that interact with and influence each other, learn from their experiences, and adapt their behavior to fit their environments better [21]. In this type of modeling, the group of autonomous agents, their associated profiles, and their environment are each represented by three parts, namely their traits and behaviors, the relationships and methods of interaction that describe how they interact, and the environment in which they interact [21].

For agent-based modeling, NetLogo 6.2.2 provided an integrated development environment for modeling. In a simulated environment, this application was used to test an enhanced user behavior model [21].

5.1 Zero Trust Principles

Using Zero Trust, implicit trust is eliminated, and entities are continuously verified based on their environment by embedding ”comprehensive security monitoring; granular risk-based access controls; and automated system security across all aspects of the infrastructure in a coordinated manner.” [22]. This is a concept and architecture that seeks to integrate diverse security services into a cohesive whole. It focuses to “minimize uncertainty in enforcing accurate, least privilege per request access decisions in information systems and services in the face of a network viewed as compromised” [30]. This enables real-time decision-making based on transaction risk profiles. Additionally, each transaction is assigned a least-privileged access level based on its perceived risk. This requires a commitment from the organization, since establishing least-privilege principles is a time-consuming process and must be completely embraced by the orga-

nization to be effective.

To be effective, the process must be fully supported by the organization. In order to achieve Zero Trust's key aims, various recommendations are recommended when developing Zero Trust architectures. NIST's Zero Trust Architecture document (Rose et al., 2020, p. 7) recommends incorporating the following tenants into an organization's security architecture while attempting to implement Zero Trust:

1. All data sources and computing services are considered a resource
2. All communication is secured regardless of network location
3. Access to individual enterprise resources is granted on a per-session basis
4. Access to resources is determined by dynamic policy
    a. Based upon the observable state of identity, application/service, and asset
    b. Optionally including behavioral and environmental attributes
5. Enterprise should monitor and measure the integrity and security posture of all owned and associated assets
    a. No asset is automatically trusted
    b. Evaluation of asset is performed as well as the identity when deciding access
6. All resource authentication and authorization are dynamic and strictly enforced before access is allowed
    a. Continuous reevaluation of trust
    b. Policy-based performance of reauthentication and reauthorization with all transactions such as "time-based, new resource requested, resource modification, anomalous subject activity detected" (Rose et al., 2020, p. 7)

Zero Trust uses the architectural tenants of the authorization model used by Extensible Access Control Markup Language (XACML), a policy language for access control decisions and enforcement. In RFC 2904, three primary components are defined: the policy engine (PE), the policy decision point (PDP), and the policy enforcement point (PEP). These components are used by administrators to make and enforce access decisions based on rules defined by administrators (Vollbrecht, et al., 2000).

Clients in this model would seek access to resources governed by PE policies and arbitrated by PDPs before being enforced by PDPs. As explained later in this paper, this high-level model is used to implement user-specific rules in agent-based modeling, where policies are implemented using code and parameters within NetLogo.

5.2 NetLogo – A modeling approach

Wilensky & Rand [37] describe NetLogo as a programming, open source, and runtime environment for modeling complex systems with many distributed parts that interact. Scalable, agent-based programming environment that combines the Logo programming language with an integrated modeling environment. The ease of implementing and evaluating NetLogo's self-organization, as well as its ability to address individual nodes in a network, makes it a powerful tool [7]. This platform allows for simple scaling by inserting parameters into the model's input. A NetLogo agent is referred to as a "turtle" that can be scaled to a large number of agents. Code and configuration are used to define how turtles interact with one another and with their surroundings. Turtles can also be divided into individual profiles known as "breeds." The "ground over which the turtles move" is a matrix of patches in which a "turtle" lives Turtles communicate with one another through links. There is an observer who aids in the coordination of the envi-

ronment's behavior. All of the behavior is represented by code written in the NetLogo programming language or by extensions written in other languages such as Python or Java. (NetLogo User Manual).

5.3 National Institute of Standards and Technologies (NIST)

It has become increasingly difficult for organizations to regulate devices that use their services, have regulations governing these operations closely, and have alternate user activities that result in a higher risk profile. As a result, the primary actors in this project are user identities and their accompanying behavior. The National Institute of Standards and Technologies (NIST) Digital Identity Rules, in particular, were used to define identity assurance guidelines for identity proofing and authentication in order to assess user interactions with an organization. Among the topics covered by the NIST recommendations are identity establishment, credential validation, and federation, which allows identities to be shared across security borders. Establishing an assurance level for identity and authentication is critical to determining the risk level for a transaction. These assurance levels are often achieved through an organization's risk assessment.

An analysis of the user's behavior in a transaction can be used to create a risk profile. This project aligned with the NIST assurance level model to accommodate more processes. We mapped to the NIST assurance level model to incorporate a proactive awareness, social networking, and browsing suspicious websites assurance level. The assurance levels attained by each of the processes determine the scores used to construct policies in this simulation. The following assurance levels were employed, which were loosely based on the assurance levels in the NIST guidelines:

1. Identity Assurance Level (IAL): The robustness of the identity proofing process

to confidentially determine the identity of an individual. (Garcia, Grassi, Fenton, 2017, p.18)

- IAL0: No identity information has been provided
- IAL1: The identity is self-asserted; no identity proofing has been
- IAL2: Knowledge-based question (DOB/Address/Last 4 SSN)
- IAL3: Picture of driver's license/Registered biometric

2. Authentication Assurance Level (AAL): The robustness of the authentication process itself, and the binding between the authenticator and a specific individual's identifier. (Garcia, Grassi, Fenton, 2017, p.18)
   - AAL0 – No credential was provided for authentication
   - AAL1 – A password or secret was provided as the authenticator
   - AAL2 – A one-time password (OTP) was provided as the authenticator
   - AAL3 – A biometric authenticator from the individual was used in the authentication process
3. Temporal Assurance Level (TAL): The risk associated with the time of transaction based upon historical usage or policy.
   - TAL0 – No timestamps or other temporal information was provided
   - TAL1 – Authentications are outside normal authentication timelines
   - TAL2 – Authentications are inside normal authentication timelines
   - TAL3 – Previous authentications match with authorized timeframes based upon policy
4. Issue-Specific Policy Assurance Level (IAL): Risks associated with visiting websites and using social media.

- LAL0 – Accessing suspicious websites and using social media
- LAL1 – accessing suspicious websites and not using social media
- LAL2 – Not accessing suspicious websites and using social media frequently
- LAL3 – Not accessing suspicious websites and using social media rarely

5.3.1 Mapping user security behavior into NIST assurance levels

In this table **??** we indicate the mapping we came up with that maps the user behaviors to the NIST assurance levels.

User Behaviors	NIST Assurance levels
Device Securement	Authentication Assurance Level (AAL)
Identity	Identity Assurance Level (IAL)
Updating	Temporal Assurance Level (TAL)
Password Generation	Authentication Assurance Level (AAL)
Proactive Awareness	Issue-Specific Policy Assurance Level (IAL)
Social Media	Issue-Specific Policy Assurance Level (IAL)
Accessing Suspicious Websites	Issue-Specific Policy Assurance Level (IAL)

5.4 Trust Algorithm

A zero trust architecture implementation uses a trust algorithm to determine whether access to a resource should be permitted or prohibited. In order to implement a policy, the policy engine collects information from multiple sources, such as subject traits and

roles, previous subject behavior patterns, threat intelligence, and other metadata [30] that were modeled in the Netlogo environment.

Depending on the algorithm, the information from these sources can be aggregated and weighted to produce a score. Below is a summary of these inputs:

1. Access Request: A request from a subject that contains information about both the user and the subject, such as the operating system, the source code, and the patch level.
2. Subject database: The subject requesting information, which includes assigned attributes and privileges that serve as the foundation for policies governing resource access. This includes the PEP's logical identity and authentication checks. The identity is made up of a set of attributes from which a level of confidence can be derived. Time and location are two examples of derived attributes.
3. Asset database: A repository for all assets and their known status. This database can be used to compare what the access request sends.
4. Resource requirements: Policies that link identity attributes like user ID and specify the bare minimum conditions for accessing an asset. , data sensitivity level and assurance level are examples of this.
5. Threat intelligence: Information on current threats and malware that is currently active, from threats that are more general in nature to specific activities that have been noticed in the environment that seem malicious.

Potential variations in trust algorithms must be taken into account because they highlight the significance of various criteria or tests. Choosing whether access decisions should be binary or weighted, as well as whether the algorithm should be score-based

or criteria-based, are all parts of this process (Rose et al., 2020). The criteria-based algorithm establishes a list of requirements that must be met before access is granted. A score-based algorithm: If the total score exceeds the threshold of an asset, a confidence level is established based on the input's weighted values, and access is granted.

Furthermore, the trust algorithm can also evaluate user requests independently or within the context of other user requests, as described below.

1. Singular algorithm: Each request is evaluated independently, and past behavior is not taken into account.
2. Contextual relationships: Access requests are evaluated based on the subject's history and the environment's history. Choosing a criteria-based/contextual trust algorithm was based on the fact that it can provide more flexible and dynamic access control since the score gives a current level of confidence regarding the account requesting access and adapts more quickly to changing elements than static policies modified by humans [30].

Based on current and previous information about the subject, this model specifically assigns a weighted overall score. The trust algorithm makes use of these scores to establish user policies. A policy regarding the user's access to a specific asset (resource) is determined using the assertion of identity, an authentication procedure, and historical and intelligence data. Assets are categorized as high, moderate, or low risk in this model. The trust algorithm will either grant or deny access based on the user's overall assurance score and the resource's criticality at the time of access.

5.5 Simulation

In the simulation, the framework is the NIST Zero Trust Access model, which describes the overall workflow for implementing Zero Trust. A subject that is presumed to be in an untrusted zone attempts to access a resource in an implicit trust zone that is guarded by an enforcement point. A policy decision point determines whether access is granted or denied based on a policy generated by a trust algorithm. Next, Users will be assigned policies based on their decisions. The enforcement of policies is determined based on user compliance with the rules. This research is an extension to the model developed by Schechinger [2].

During the start step, a user-represented subject attempts to access a resource in the implicit trust zone (at the AZ step). The transaction needs to be approved by the policy decision point before the user can access the resource. This is demonstrated by running a number of tests using the user's identity, authentication, temporal, and issue-specific behavior. Access is granted or denied based on this data and the policy that the trust algorithm generates. In the last step, Based on their decisions, policies will be assigned to users.

A number of tests are performed as part of the policy decision, including operations for identifying, authenticating, and determining the state of the user, such as:

- Establishment of the identity through one of four identity assurance levels
- Authentication of the user based upon the strength of the authentication of the user through one of four authentication assurance levels
- Measurement of issue-specific parameters to determine how well users interact online
- Time-based measurements (temporal) are taken in order to determine whether a

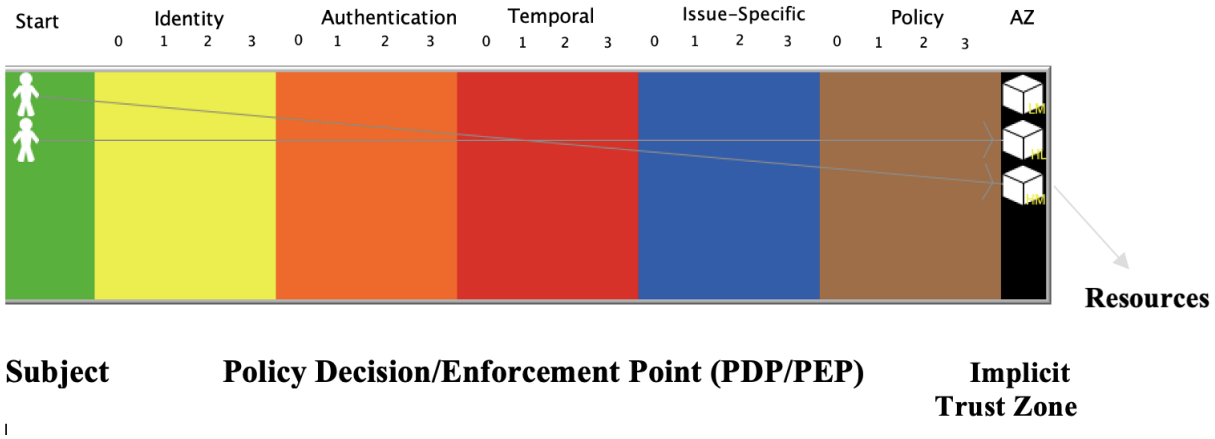


Figure 5.1: Simulation of Zero Trust model

user's activity corresponds to their previous access to the environment

Depending on the aggregate scores of the users, a score and corresponding policy are provided for granting access to assets corresponding to their severity-level (high, moderate, or low). In each simulation run, the user specifies the number of iterations, and both authorization successes and failures are recorded. Raw scores are generated according to the diagram below.

An administrator supplies an adjustable number of users and assets at the start of the model (turtles representing the corresponding breeds of "user" and "asset"). Each user starts in the Start region, and each asset is assigned to the AZ region, which represents the implicit trust zone. Each asset is given a device as well as a network criticality level of low, moderate, or high. Each user is assigned to a different asset at random (to simulate a user attempting to access the resource).

As the simulation begins, the user will move through phases related to identity, authentication, temporal, and issue-specific phases. A random assurance level is assigned to users during each phase based on the weighting provided in the assurance level sliders

Assurance Level	Test Criteria	Score
IAL0	No identity information provided	0
IAL1	The identity is self-asserted	1
IAL2	Knowledge-based question	2
IAL3	driver's license/Registered biometric	3
AAL0	No credential	0
AAL1	password or secret	1
AAL2	OTP	2
AAL3	Biometric	3
TAL0	No timestamps	0
TAL1	Authentications are outside normal	1
TAL2	Authentications inside normal timelines	2
TAL3	Authentication within authorized timeline	3
LAL0	suspicious websites / social media	0
LAL1	accessing suspicious not social media	1
LAL2	Not accessing suspicious/ social media	2
LAL3	Not accessing websites / media rarely	3

Figure 5.2: Assurance level scoring

Figure 5.3: Counters for users and assets

(administratively set levels based on NetLogo sliders).

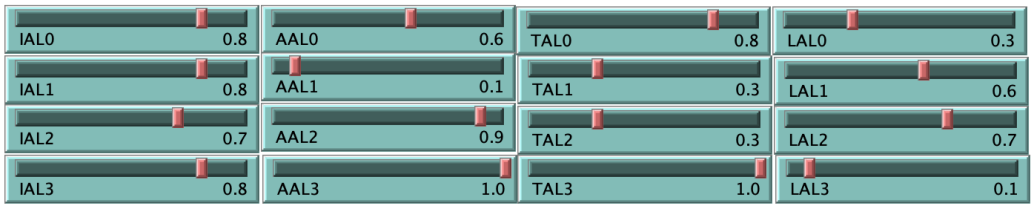


Figure 5.4: Assurance Level weighting

After the model has collected (produced) all of the assurance levels, a trust algorithm is used to determine whether the generated scores have permission to access the resource. If the aggregated score meets the following requirements, the trust algorithm (decided by the ”is-authorized” procedure) grants access:

Criticality	Aggregate Score
Low	6
Moderate	4
High	2

Figure 5.5: Asset to score mapping

If the algorithm determines that a given score is sufficient for the user to access the asset, the asset will turn green. The number of successful authorizations will be indicated by an incrementing counter. The asset, however, turns red if the user isn't authorized to access it. As the model executes, two separate counters are incremented

in order to identify the number of unsuccessful and successful authorizations. A graph will be drawn summarizing the number of successful and unsuccessful authorizations.

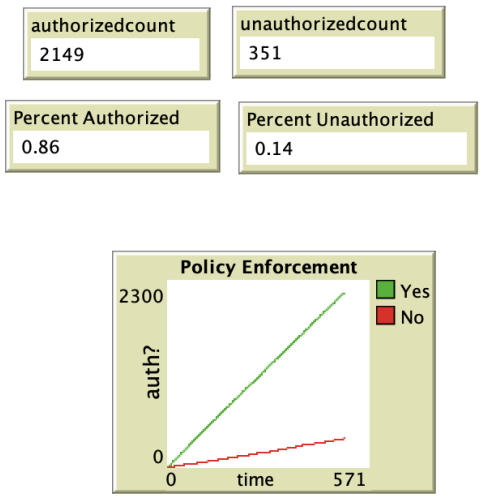


Figure 5.6: Authorization summary

Based on the security practices of specific users, policies were created to impose on them. Users' decisions determine the policies that will be implemented, which show their adherence to the assurance level.

- Most Strict Policy (0): When the user exhibits extreme disregard for the security measures required.
- Strict Policy (1): When the user exhibits disregard for the security measures required.
- Moderate Policy (2): When the user follows security procedures without following the required procedures in a slightly irresponsible manner.
- Restricted Policy (3): In this case, the user adheres to appropriate security measures while demonstrating responsible security behavior.

The policies are imposed on users according to the following criteria:

Policy	Aggregate score
0	>= 3
1	>=7
2	>=11
3	>=15

Figure 5.7: Policies Criteria

To compare different criteria for different tests, user-supplied parameters included how many iterations each model would execute, how many users and devices were used to start the simulation, and how assurance levels were weighted according to asset criticality levels. As assurance levels and criticality levels are application variables, this paradigm can be easily extended.

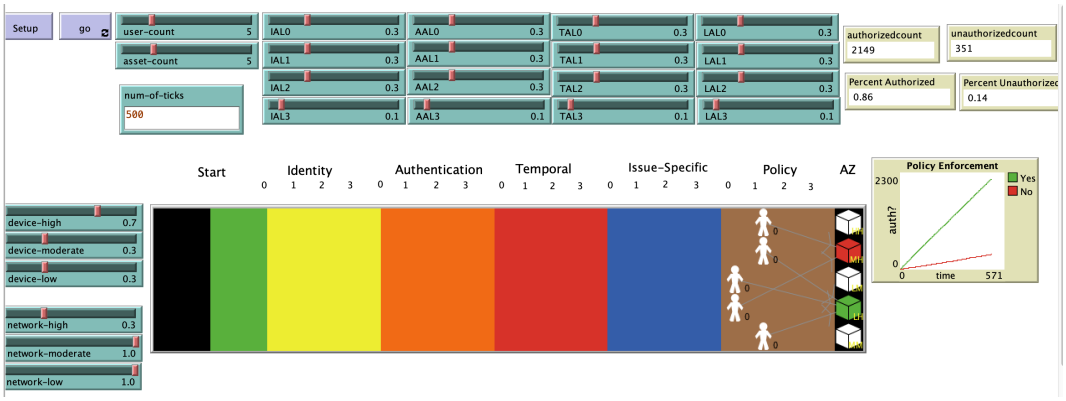


Figure 5.8: NetLogo (Zero Trust Model)

Chapter 6

Conclusion

In this paper, we used Uppaal to model different user security behaviors and further explored whether agent-based modeling can be extended to model enhanced user behavior in a Zero Trust environment. Furthermore, this paper maps user security behavior to NIST assurance levels. It appears that the extended research on Zero Trust principles and architecture aligns with NIST's identity guidelines. The model simulated the security behaviors of end users in NetLogo to generate policies. We were able to address the concept of Zero Trust and eliminate the trust that all users act responsibly.

Most importantly, we had some success responding to the research question by addressing the Zero Trust concept and eliminating the assumption that users act responsibly.

Q1: What is the possible extension to end-user security behaviors?

This research extended user security behavior to include social media, suspicious websites, and identity as security behaviors. As well as extend password generation behavior to include one time password.

Q2: How can we map the end user behavior to NIST assurance levels? Additionally, addressing the complexities when modeling and simulation environment is different?

We mapped user security behavior to NIST assurance levels. We expanded the NIST assurance level model to include issue-specific policy assurance levels mapped to proactive awareness, social media, and website access.

Q3: How to implement security policies in an agent-based environment?

The appropriate policy was given to address security flaws generated by a specific user after observing and evaluating security behavior.

This research can be extended to develop an automated mapping from the Uppaal behaviors to the Netlogo implementations. Additionally, other behaviors can be included that were identified from the NIST document, but were not implemented in this research.

Bibliography

[1]

[2] Agent based modeling and simulation for user specific policy, 2021.

[3] A. Adams and A.S. Sasse. Tusers are not the enemy. commun. acm 42. 73, 1999.

[4] Alana Maurusha Ahmed A. Moustafa, Abubakar Bello. The role of user behaviour in improving cyber security management. *Frontiers in Psychology*, 73, 2021.

[5] Arwa AlQadheeb, Siddhartha Bhattacharyyaa, and Samuel Perlr. *Enhancing Cybersecurity by Generating User-Specific Security Policy through the Formal Modelling of User Behavior*. 2022.

[6] R. Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1999.

[7] Magula P. Babis, M. Netlogo — an alternative way of simulating mobile ad hoc networks. 2012 5th joint ifip wireless and mobile networking conference (wmnc), 2012.

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

[9] Kayce Basques. Why https matters, 2020.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] North M. J. Macal, C. M. utorial on agent-based modeling and simulation. journal of simulation. 2010.

[22] North M. J. Macal, C. M. National security agency. embracing a zero trust security model., 2021. doi:, https://media.defense.gov/2021/Feb/25/2002588479/-1/-1/0/CSI_EMBRACING_ZT_SECURITY_MODEL_UOO115131-21.PDF.

[23] Microsoft. The dangers of oversharing. https://support.microsoft.com/en-us/topic/the-dangers-of-oversharing-79330a32-4ee1-433a-812e-fe4bb3d34511#:~:text=Sharing%20pictures%20of%20your%20home,or%20potentially%20make%20them%20targets., 2022.

[24] 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, 2008.

[25] NordLayer. What are malicious websites and how can you identify them?, 2022.

[26] Rami Radwan Omar and Tawfig M. Abdelaziz. A comparative study of network access control and software-defined perimeter. in proceedings of the 6th international conference on engineering mis 2020 (icemis'20)s, 2020.

[27] 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.

[28] 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.

[29] D.B. Parker. estating the foundation of information security. in g.c.gable and w.j. caelli eds. it security: The need for international co-operation.2., 1991.

[30] Borchert O. Mitchell S. Rose, S. Zero trust architecture., 2020. doi: ttps://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf.

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

[32] TechTarget. What is knowledge based authentication (kba)? https://www.techtarget.com/searchsecurity/definition/knowledge-based-authentication, 2010.

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

[34] 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.

[35] 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.

[36] 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.

[37] U. Wilensky and W. Rand. *Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo.* MIT Press, 2015.

[38] Wang H. Yan X. Survey on zero-trust network security. in: Sun x, wang j, bertino e, editors. in: Communications in computer and information science. artificial intelligence and security, 2020.

[39] Kaliya Young. The identity spectrum, 2010.

[40] 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.