[Array 14 (2022) 100146](https://doi.org/10.1016/j.array.2022.100146)

|  |  |  |
| --- | --- | --- |
|  | Contents lists available at [ScienceDirect](http://www.elsevier.com/locate/array) |  |
| Array |
| journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array) |
|  | | |

Enhancing cybersecurity by generating user-specific security policy through the formal modeling of user behavior   
Arwa AlQadheeba, Siddhartha Bhattacharyyaa,∗, Samuel Perlb   
a *Computer Engineering and Sciences, Florida Institute of Technology, Melbourne, FL, 32901, USA*   
b *Software Engineering Institute , Carnegie Melon University, Pittsburg, PA, 15213, USA*

|  |  |  |
| --- | --- | --- |
| A R T I C L E | I N F O | A B S T R A C T |
| *Keywords:*  Zero Trust  Automatic verification  Correctness  Cybersecurity policy  Security policy  Formal methods  User behavior  Automated security policy generation Finite-State Automata  Timed Computation Tree Logic | | Organizations today are faced with the difficult challenge of balancing the embrace of new and emerging technology, and securing their systems and data that support critical business functions. Although there have been significant advances in security enforcement technology, attackers are still able to compromise organiza-tions and access. The impacts of computer intrusions have become so untenable that many organizations are looking at a drastic rethinking of their approach to the security of internal networks. This approach is called *Zero Trust* and it seeks to remove all notion of a trusted internal network boundary. The benefits of *Zero Trust* include significantly increasing the work that attackers would need to perform to achieve their objectives. But *Zero Trust* will also increase the management complexity for internal security teams. These teams will need a way to collect data and enforce policy decisions based upon analysis. This process will need to be done for all organizational systems, and data, and it will need to be done in all access contexts.  Our approach uses formal methods to model and examine end-users security-related behaviors. Researchers have found that the users’ security decisions correlate with factors including demographics, personality traits, decision-making styles, and risk-taking preferences. We describe these behaviors by using Finite-State Automata (FSA). This allows for the automated formulation of linear-time security properties based on Timed Computation Tree Logic (TCTL). The logic is then used to check the satisfaction of collected and observed security behaviors against policy. This formal behavioral analysis could be combined with other security and network data during the context analysis process that needs to occur for each *Zero Trust* access request. Other network or host security data could include address identifiers, tokens, event data, packet inspection, running process data, cyber threat intelligence, and much more. Our method allows organizations that embrace a *Zero Trust* philosophy to generate context specific security policies that can be automatically verified for correctness and completion. |

**1. Introduction**

Organizations today are faced with the difficult challenge of bal-ancing the embrace of new and emerging technology, and securing their systems and data that support critical business functions. Al-though there have been significant advances in security enforcement technology, attackers are still able to compromise organizations and access. Perhaps more now than ever before, Organizations are re-quired to embrace new technical innovations including advances in the Internet of Things, Machine Learning and Statistical Analysis, and Cloud Computing. As these new devices and services become available, organizations have a business need to connect them to their business networks. If they do not, they risk becoming overtaken in their market by competitors. This further tests their security teams ability to prevent

∗ Corresponding a[uthor.](mailto:sbhattacharyya@fit.edu)

*E-mail address:* [sbhattacharyya@fit.edu](mailto:sbhattacharyya@fit.edu) (S. Bhattacharyya).

new intrusions because the attack surface changes quickly and the software supply chain becomes ever more complex.

Security professionals are developing new security models to be better prepared to Prevent, Detect, Respond, and Recover from cy-berattacks. In 2010, John Kindervag, a Security and Risk Principal Analyst at Forrester Research Inc., developed a *Zero Trust* security model that radically prioritizes a classic security principles; ‘‘never trust, always verify’’ [1]. The rules of operation for how a *Zero Trust* network operates are different from traditional security models. They reflect lessons learned from over a decade of dealing with intrusions and observing the adversaries tactics, techniques, and procedures. Organizations considering *Zero Trust* will also need to pay atten-tion to the human behaviors [2]. The natural structure of humans

<https://doi.org/10.1016/j.array.2022.100146>  
[Received 10 June 2021; Received in revised](https://doi.org/10.1016/j.array.2022.100146) form 15 February 2022; Accepted 1 April 2022 Available online 9 April 2022

2590-0056/© 2022 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

psychology, the limitation of humans’ information processing capacity, and their reliance on previous experiences to take future action can impede their security choices [3]. The Ponemon Institute study based on interviewing 507 companies across the globe reveals that accidents among users is the leading cause of 24% of data breaches, which are worth, on average, around $3.5 million of financial damage. Uninten-tional 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 takes organizations around 181 days and 61 days to identify and contain, respectively, a data breach that is related to such careless human action [4].

Most computer systems are designed based on the general percep-tion of that all users in an access group will behave similarly. This overlooks the idea that individuals security behaviors differ from each other [5]. Researchers in human factors, have examined the psychology of humans to understand users’ different behaviors toward privacy and security. Some works show 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. One research study found Norwegians and Japanese tend to browse the Internet more cautiously than their counterparts in Italy and Spain, thus lower their chances of randomly encountering malware-infected websites [6].

The prominent hacker Kevin D. Mitnick, states that the most effec-tive 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 [7]. 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 generate user specific policies based upon security behaviors measured with scales in existing academic literature studies. The question we address in our research is:   
**Problem Statement** How to generate explicit security policy after observing and analyzing end user security behaviors along with other assumed *zero trust* security enforcement decisions?’’  
 Our method demonstrates the use of Formal Method based frame-work for generating user specific security policy that suggests a new contribution to the extant literature.

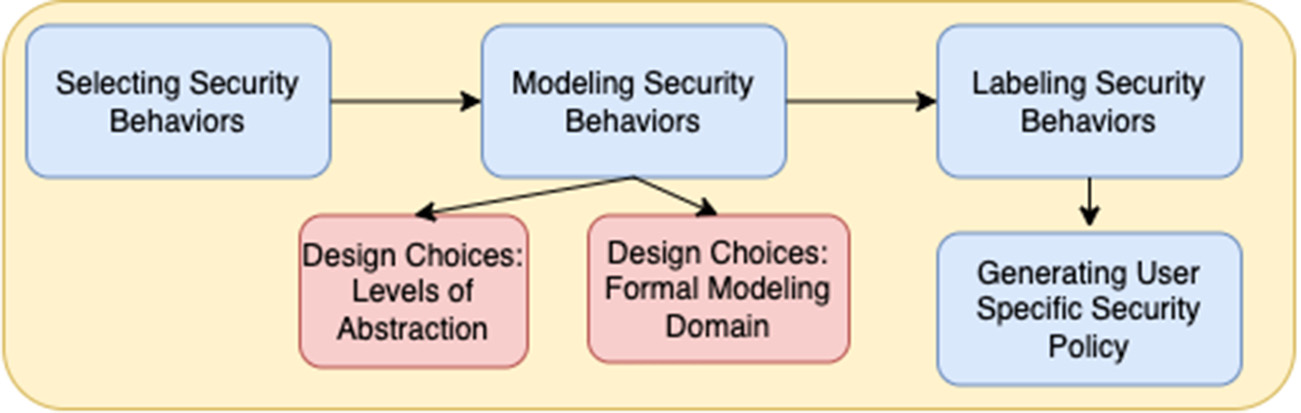
The remainder of this research is organized as follows: Section 2 describes in greater depth related work. Section 3 presents the research methodology, outlining the sequential development process for the automated approach. Section 4 discusses the selection of user secu-rity behavior. Section 5 discusses the model considerations and the selected modeling paradigm. Section 6 discusses the overall experiment performed with selection and modeling of the security behaviors in Section 6.1, Section 6.2 explains the checking of satisfaction of user security behavior. Section 6.3 defines the rule of generating a user-specific policy. Section 7 discusses the final results obtained by using our framework. Finally, Section 8 states our conclusions and future work.

**2. Related work**

Formal method based approach has been applied to the modeling and analysis of user behavior, with user models modeled in cognitive architecture, as described by Curzon [8]. With this approach the re-searchers were able to capture potential erroneous interactions between devices and humans. The focus of the research by Curzon was in the

2

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*



**Fig. 1.** Research methodology.

Our approach uses results from the study of human intentions, at-titudes, norms, and resulting cybersecurity decision making behaviors. There are many studies showing that usable security is a hard problem. Users often ignore explicit warnings and misunderstand the likelihood of many risks and their resulting impacts [17]. Users also rationally reject warnings because they have experienced so many false alarms in their past experience [18,19]. Human and Computer interaction interfaces are hard to design and inherent cognitive weaknesses can negatively affect security decisions [3,20]. Some studies find that indi-vidual users behave differently when presented with security decisions based upon background, culture, and attitudes [2,5,21].

Because human behaviors are varied, difficult to predict, and open ended, we limit the security behaviors in our study to those with an existing scale that has been validated in previous studies. Egelman and Peer created a Security Behavior Intentions Scale (SeBIS) [12] to evaluate users’ ability to adhere to computer security advice based on self-declared information. The scale focuses on four constructs which are: device securement, password generation, proactive awareness, and updating. Gratian et al. [2] substantiate the accuracy of SeBIS and broaden SeBIS to examine the correlations of personality traits and demographics with security intentions.

Our Formal Method guided User-Specific Policy Generation Frame-work (FMUSPGF) in Fig. 1 sets out the structure of this research as a whole, the figure shows the development process as a sequence of four stages.

The first step in FMUSPGF involves selecting security behaviors, this was achieved by conducting a survey of existing literature to identify what are the expected security behaviors. Once the security behaviors are selected, the second step involves modeling these behaviors, where we implemented a formal method-based approach. After the security behaviors are modeled, the next step involves labeling the security behaviors, this was accomplished by executing queries designed us-ing formal specification. Finally, user specific security policies are generated guided by the labeling in the previous step.

**Our main contribution** is in the design of a formal method based framework to support the process of user specific policy generation for cybersecurity. In the process, we demonstrated how to identify and select the essential characteristics that define users security behavior. Then, we modeled the identified security behaviors as formal models to enable automated reasoning. Finally, we were able to detect weakness in users security behavior and then propose relevant policies.

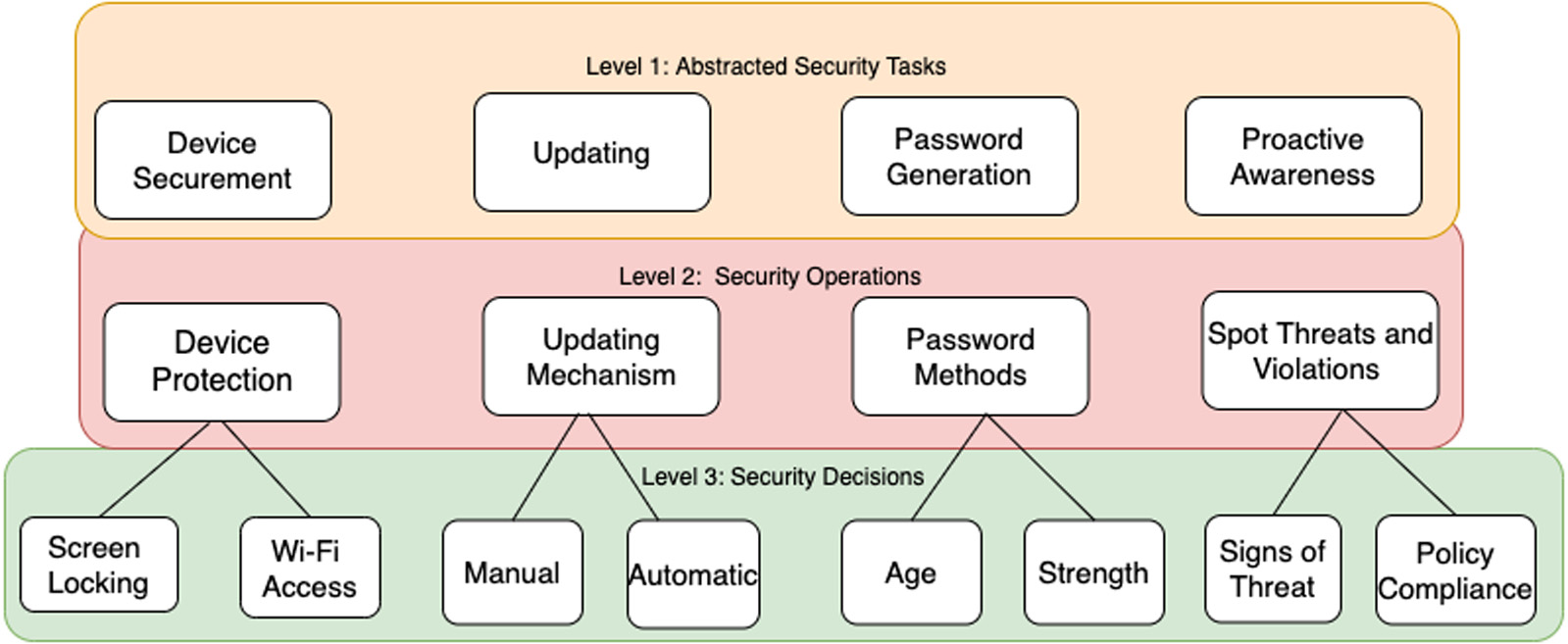
**4. Selecting security behaviors**

In this section, we focus on selecting user’s security-related behavior by taking into account the 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 represents the require-ments for developing the formal model, which we accomplished after reviewing a wide range of possibilities.

Previous studies have comprehensively investigated the correlations between individual differences in demographics, personality traits, decision-making styles, and risk-taking preferences and their influence

3

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*



**Fig. 2.** Cybersecurity user behavior knowledge representation architecture.

**5. Modeling user security behavior**

After reviewing studies on how the decision-making process related to security behavior, the appropriate knowledge base was built, and the most relevant user behaviors were selected to develop the formal model. The content of this section focuses on the design choices that were made pertaining to the architecture with levels of abstraction, and the modeling paradigm that was selected.

*5.1. Design choice: Levels of abstraction*

It is challenging to model human–machine interactions, because of the complexity of human behavior and the broad set of knowl-edge requirements. Although, we chose a specific knowledge base, it is still challenging to model and examine every aspect of a users security-related behavior that is relevant to device securement, pass-word generation, proactive awareness, and updating. To model users behavior across the different aspects of SeBIS, we applied principles of decomposition inherited from software/systems engineering to ar-chitect the structure of the model. the architecture of the model is as shown in Fig. 2.

The architecture modeling the representation of knowledge about the security behavior includes different levels of abstractions, in order to ease the debugging, increase readability/maintainability and lessen the complexity. We decomposed the structure into different sorts of se-curity services on multiple layers, starting from **(1)** as the highest level of abstraction and ending with **(3)** 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.

Here, we illustrate each level of abstraction in detail.

• **Layer (1)** is the most abstract of the four security service check layers. It assimilates the SeBIS concepts: device securement, pass-word generation, proactive awareness, and updating that are specified by Egelman and Peer. These correspond to the highest level of representation of the security task that needs to be executed.

• **Layer (2)** breaks down SeBIS concepts according to key param-eters within each category as identified at Level 1 for example, device protection, update mechanism, password methods, and attention to threats and violations. Then, it is further decomposed into possible additional sub-services based on the user decisions on the actions they want to take.

• **Layer (3)** has a sub-tree that descends from the specifics of Layer 2 for example screen-locking feature for device protection and is enabled by what the user chooses.

4

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

action, a guard and a set of clocks to be reset; and *𝐼* ∶ *𝐿* → *𝐵*(*𝐶*) assigns invariants to locations.

We define a clock valuation as a function *𝑢* ∶ *𝐶* → R≥0 from the set of clocks to the non-negative reals. Let R*𝐶*be the set of all clock valuations. Let *𝑢*0(*𝑥*) = 0 for all *𝑥* ∈ *𝐶*. If we consider guards and invariants as the sets of clock valuations (with a slight relaxation of formalism), we can say *𝑢* ∈ *𝐼*(*𝑙*) means *𝑢* satisfies *𝐼*(*𝑙*).

• **Timed Automata Semantics**   
 Let (*𝐿, 𝑙*0*, 𝐶, 𝐴, 𝐸, 𝐼*) be a timed automata *𝑇 𝐴*. The semantics of the *𝑇 𝐴* is defined as a labeled transition system ⟨*𝑆, 𝑠*0*,* →⟩, where 1. (*𝑙*, *𝑢*) → *𝑑* (*𝑙*, *𝑢* + *𝑑*) if ∀ *𝑑*′: 0 ≤ *𝑑*′≤ *𝑑* ⇒ *𝑢* + *𝑑*′∈ *𝐼*(*𝑙*) 2. (*𝑙*, *𝑢*) → *𝑎* (*𝑙*′, *𝑢*′) if ∃ *𝑒* = (*𝑙*, *𝑎*, *𝑔*, *𝑟*, *𝑙*′) ∈ *𝐸* such that *𝑢* ∈ *𝑔*, *𝑢* = [*𝑟* ↦ 0] *𝑢* and *𝑢*′∈ *𝐼*(*𝑙*)

where for *𝑑* ∈ R≥0, *𝑢* + *𝑑* maps each clock *𝑥* in *𝐶* to the value *𝑢*(*𝑠*) + *𝑑*, and [*𝑟* ↦ 0]*𝑢* denotes the clock valuation which maps each clock in *𝑟* to 0 and agrees with *𝑢* over *𝐶* ⧵ *𝑟*.

Note that a guard *𝑔* of a *𝑇 𝐴* is a simple condition on the clocks that enable the transition (or, edge *𝑒*) from one location to an-other; 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 transi-tion on edge executes. Thus, a timed automata 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 *𝑇 𝐴𝑖* = (*𝐿𝑖, 𝑙𝑖*0*, 𝐶, 𝐴, 𝐸𝑖, 𝐼𝑖*), 1 ≤ *𝑖* ≤ *𝑛*. This enables us to check reachability, safety, and liveness properties, which are expressed in temporal logic ex-pressions, over this network of timed automata. An execution of the *𝑇 𝐴*, denoted by *𝑒𝑥𝑒𝑐*(*𝑇 𝐴*) is the sequence of consecutive transitions, while the set of execution traces of the *𝑇 𝐴* is denoted by *𝑡𝑟𝑎𝑐𝑒𝑠*(*𝑇 𝐴*).

*5.3. Query language for verification*

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) [31]. The syntax of the Timed Computation Tree Logic is expressed as follows:   
 *𝛷* ∶∶= *𝑎* | *𝑔* | *𝛷*1 ∧ *𝛷*2 | ¬ *𝛷* | *𝐸* (*𝛷*1 ∪*𝐽𝛷*2) | *𝐴* (*𝛷*1 ∪*𝐽𝛷*2), 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.

• state *𝑠𝑖 ⊧* E(*𝛷*1 ∪*𝐽𝛷*2) ‘‘for some path’’ *𝑠𝑖*, *𝑠𝑖*+1... (∃ k ≥ *𝑖*, k-i ∈ J) ((*𝑠𝑘 ⊧* q) (∀ j, i ≤ j < k)(*𝑠𝑗 ⊧* p))• state *𝑠𝑖 ⊧* A (*𝛷*1 ∪*𝐽𝛷*2) ‘‘for every path’’ *𝑠𝑖*, *𝑠𝑖*+1... (∃ k ≥ i, k-i ∈ J) ((*𝑠𝑘 ⊧* q) (∀ j, i ≤ j *<* k)(*𝑠𝑗 ⊧* p))

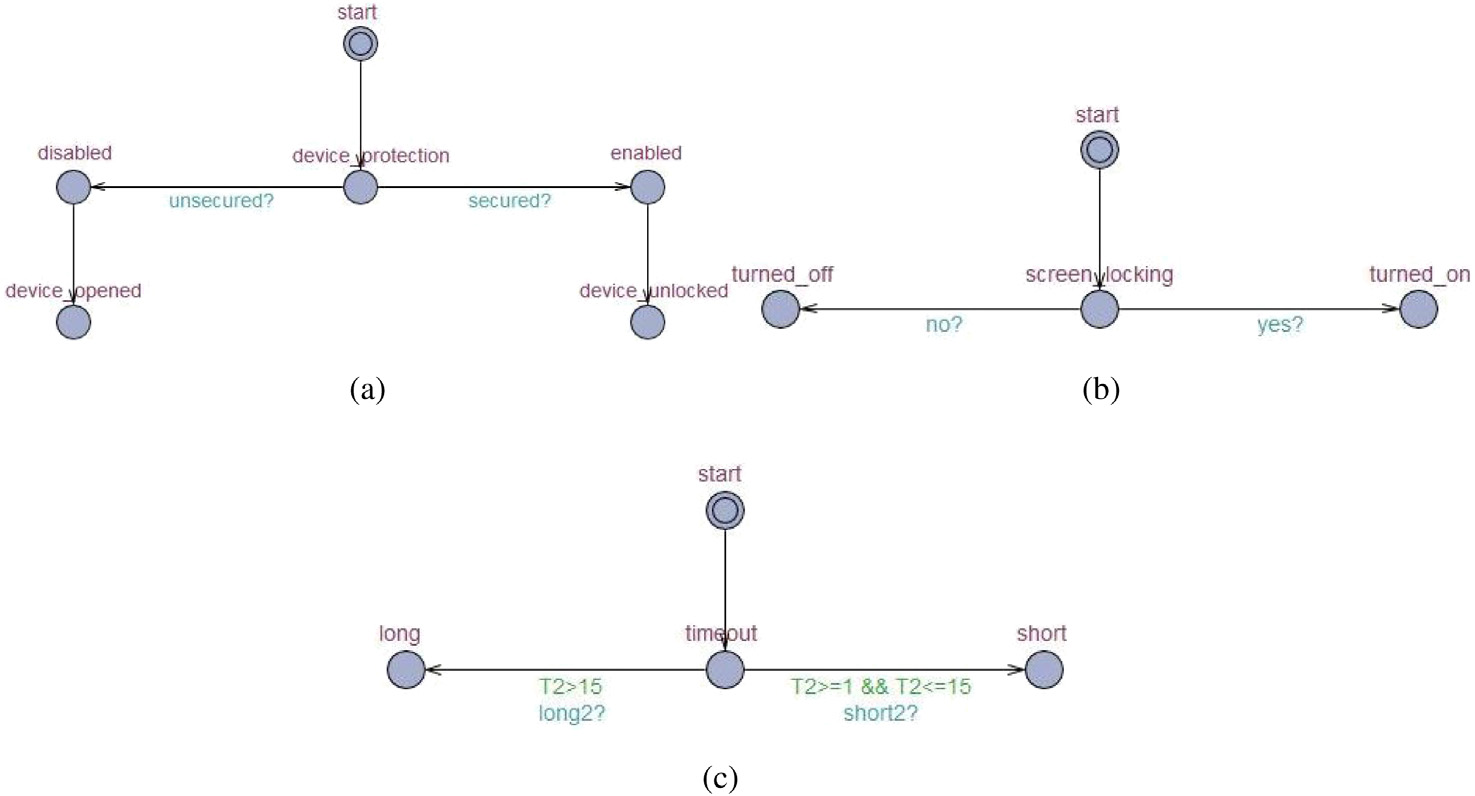
TCTL is similar to CTL in having temporal connectives that are expressed as pairs of symbols. Such that, the first element of the pair represents one of the path quantifiers that is either **A** or **E** whereas the second element of the pair is one of the state quantifiers that is one of the following:

• **G** means ‘‘all states in a path’’.  
• **F** means ‘‘some state in a path’’.

The different combinations of path formulae and state formulae accepted by Uppaal are: **AG** invariantly A[], **EG** potentially always E[], **AF** eventually A*<>*, **EF** possibly E*<>*.

5

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*



**Fig. 3.** Device securement state-transition graphs.

|  |  |
| --- | --- |
| **Table 2**  Device securement properties. | |
| No. | Knowledge base property |
| 1  2  3  4  5  6 | E*<>* (device\_protection.enabled) E*<>* (device\_protection.disabled) E*<>* (screen\_locking.turned\_on) E*<>* (screen\_locking.turned\_off) E*<>* (timeout.short)  E*<>* (timeout.long) |

• **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 [33]. Some others do set a password-protected screen saver, but they adjust the default timeout time (i.e., often 15 min) to a much longer time [34]. 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 [35]. 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 2, we can see how we translate the criteria mentioned above into TCTL formulate whereas Fig. 3 depicts device protec-tion, screen-locking, and screen-locking timeout state-transition graphs, respectively.

*6.1.2. Password generation*  
 • **Password Age**   
 Passwords are a perennial problem 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 [36]. Some might argue that scheduled changes make the passwords harder

6

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

|  |  |
| --- | --- |
| **Table 3**  Password generation properties. | |
| No. | Knowledge base property |
| 1  2  3  4  5  6 | E*<>* (password\_age.non\_expired) E*<>* (password\_age.expired)  E*<>* (password\_length.long)  E*<>* (password\_length.short)  E*<>* (password\_reusability.unused) E*<>* (password\_reusability.used) |

Table 3 lists the equivalent expressions of password age, length, and re-usability in Uppaal specification language.

*6.1.3. 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 [41]. 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 [42]. It is exactly the case in organizations’environments; reporting possible security incidents can save valu-able crucial time in the early stages of breach detection [43]. 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. The idea of including this area of interest to our research is to model 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 confidential-ity, 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 [44]. Security policies and guidelines are put in place to draw a line for employees between what is acceptable and unacceptable to do when they interact with the information sys-tem. The problem lies in the employees’ non-compliance [45]. According to Mutlaq et al. (2016), non-compliant behavior can be

7

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

|  |  |
| --- | --- |
| **Table 5**  Updating properties. | |
| No. | Knowledge base property |
| 1  2  3  4  5  6 | E*<>* (updating\_mechanism.automatically) E*<>* (updating\_mechanism.manually)  E*<>* (time\_to\_update.short)  E*<>* (time\_to\_update.long)  E*<>* (time\_to\_reboot.right\_away)  E*<>* (time\_to\_reboot.after\_awhile) |

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 [48]. 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 5, we present the formal specifications of updating that are expressed in terms of timed temporal logic TCTL.

*6.2. Experiments: Classifying security behavior*

In this section, we model several test cases with different security behaviors, as Finite-State Automata (FSA). For each separate test case, we generate a set of user-specific linear-time properties. We then clas-sify the behavior as good or bad based on the results of the reachability analysis. Once the behavior is classified we generate the relevant security permissions such as, Strict, Moderate and Least restrictive.

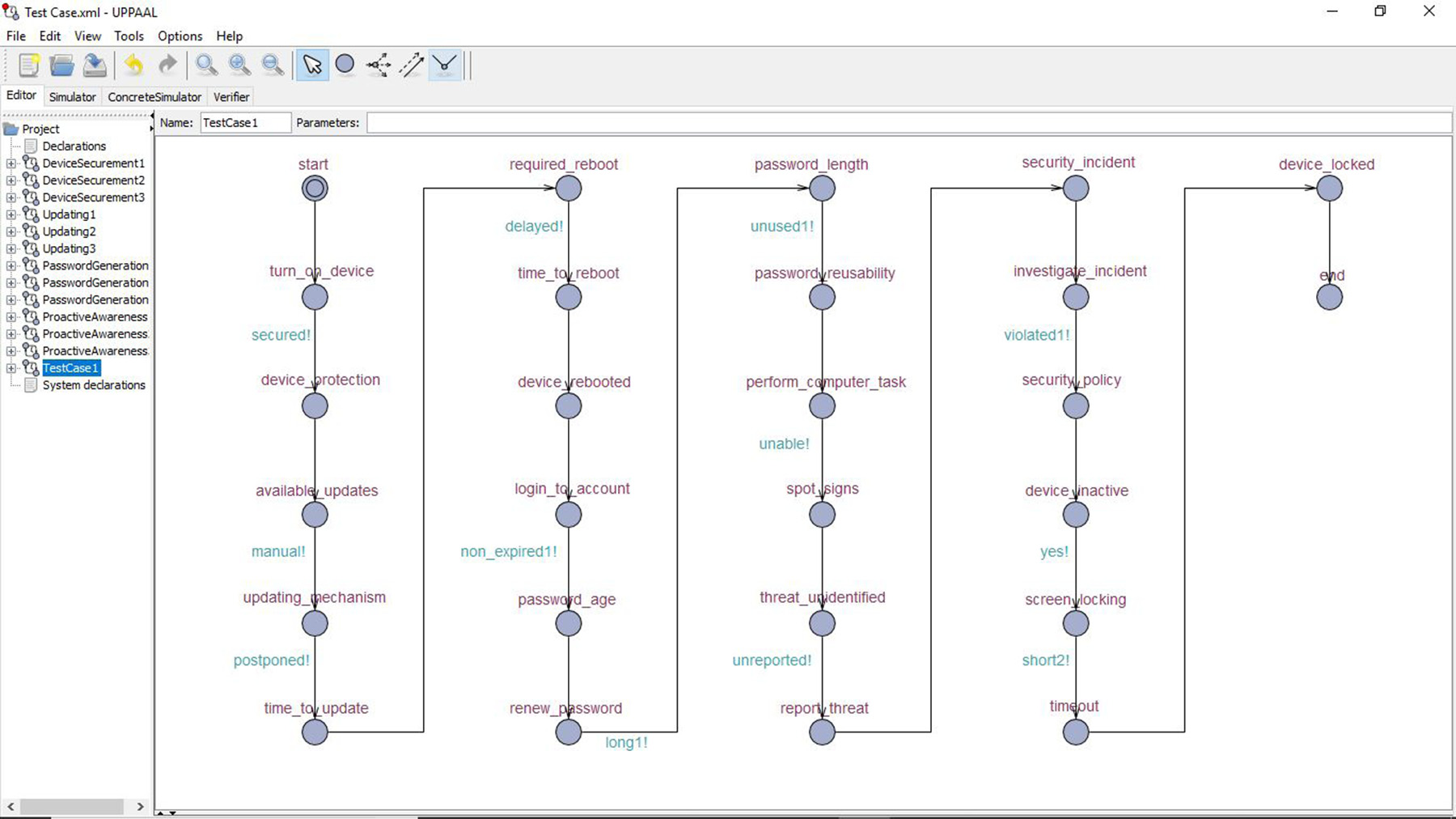
*6.2.1. Automated analysis*   
 We seek to analyze users’ behaviors in order to make a careful analysis and draw out their poor security decisions that have a signifi-cant impact on the security system. To ultimately achieve our goal, we designed six test cases that cover as much as possible of different se-curity 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,1manually generate user-specific linear-time properties, apply reachability analysis, identify good and bad security behaviors and generate user-specific policy. In this section, we demonstrate one example of these test cases as an illustration of our formal method-based approach.

*6.2.2. Representing test cases*   
 In order to have reliable test cases, we had to make several as-sumptions 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 com-pany 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 role. Fig. 4 represents the state-transition graph of Tom’s security-related behavior. All the other models can be found at [https: //github.com/sbhattacharyya/USPGZeroTrust](https://github.com/sbhattacharyya/USPGZeroTrust).

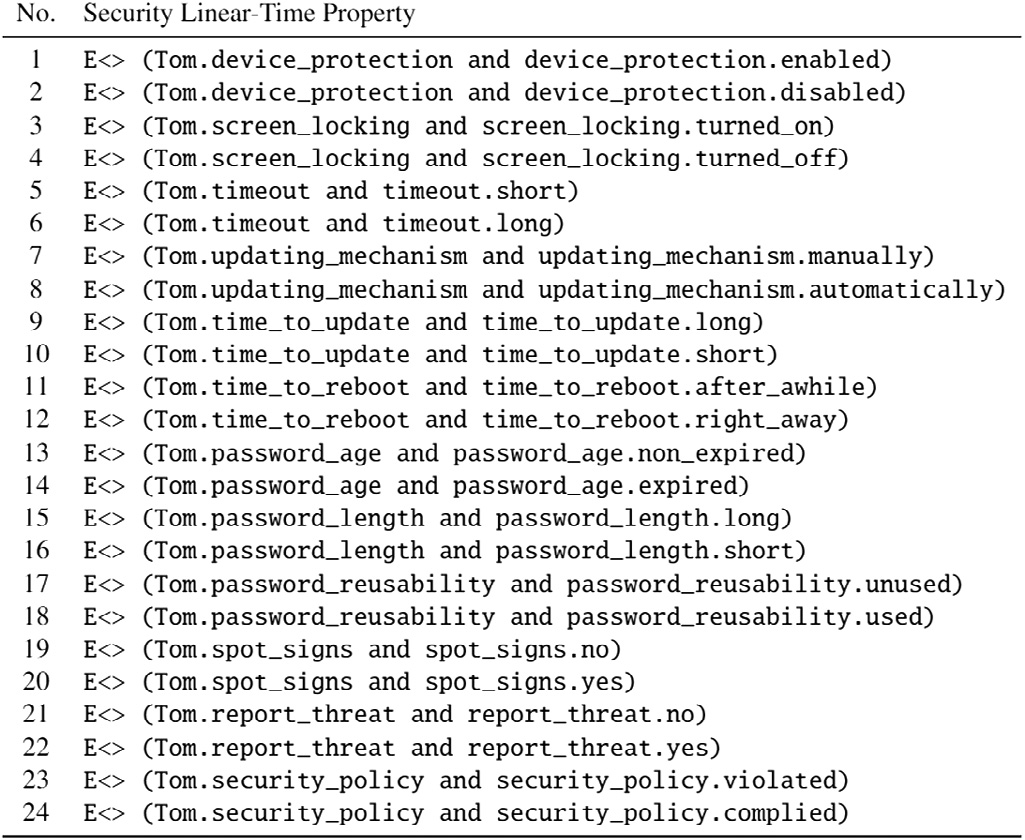
1 Integrated tool environment for modeling, validation, and verification of Finite-State Automata (FSA) [24].

8

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*



**Fig. 4.** Tom’s state-transition graph.



**Fig. 5.** Security properties verified.

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

2. **Moderate Policy** (*𝑚*): Applied when the user is exhibiting neg- ligence in following the appropriate security measures.

3. **Least Restrictive Policy** (*𝑟*): Applied when the user is exhibit-ing good security behavior with full respect to the appropriate security measures.

The assignment of a policy to a user is as described in the policy generation Algorithm 1. The input to the algorithm are: 1. model of all the known security behaviors, as TA, 2. representation of user security behaviors (*𝑈𝑖𝑆𝑏𝑘*) as a trace in TA, where *𝑈𝑖* is user i and *𝑠𝑏𝑘* is the security behavior k exhibited by user i, formulation of security proper-ties in TCTL *𝑆𝑃 𝑟𝑜𝑝𝑗*, it is the security property that is being verified and the satisfaction output, for a user exhibiting all the behaviors stored in a list *𝑆𝑎𝑡*\_*𝑂𝑢𝑡𝑖*, it stores the security behavior and the behavior label

9

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

|  |
| --- |
| **Algorithm 1**  **INPUT**: Results of Model checking performed on  Timed Automata representation of User Security Behavior (TA,  {{*𝑈*1 *𝑆𝑏*1, ..., *𝑈*1 *𝑆𝑏𝑘*...*𝑈*1 *𝑆𝑏𝑝* }, {*𝑈𝑖 𝑆𝑏*1, ..., *𝑈𝑖 𝑆𝑏𝑘*...*𝑈𝑖 𝑆𝑏𝑝*  },{*𝑈𝑛 𝑆𝑏*1, ..., *𝑈𝑛 𝑆𝑏𝑘 𝑈𝑛 𝑆𝑏𝑝* }} {*𝑆𝑃𝑟𝑜𝑝*1, ..., *𝑆𝑃𝑟𝑜𝑝𝑗*,... *𝑆𝑃 𝑟𝑜𝑝𝑚*},  {*𝑆𝑎𝑡*\_*𝑂𝑢𝑡*1...*𝑆𝑎𝑡*\_*𝑂𝑢𝑡𝑖*...*𝑆𝑎𝑡*\_*𝑂𝑢𝑡𝑛*}}  **OUTPUT**: User Specific Security Policy ((*𝑈𝑠𝑒𝑟*1, (*𝑆𝑏*1, *𝑃𝑡𝑦𝑝𝑒*11),  (*𝑆𝑏*2,*𝑃𝑡𝑦𝑝𝑒*12), (*𝑆𝑏𝑝*, *𝑃𝑡𝑦𝑝𝑒*1*𝑝*), (*𝑈𝑠𝑒𝑟𝑖*, (*𝑆𝑏*1, *𝑃𝑡𝑦𝑝𝑒𝑖*1), (*𝑆𝑏*2,*𝑃𝑡𝑦𝑝𝑒𝑖*2), (*𝑆𝑏𝑝*,  *𝑃𝑡𝑦𝑝𝑒𝑖𝑝*), (*𝑈𝑠𝑒𝑟𝑛*, (*𝑆𝑏*1, *𝑃𝑡𝑦𝑝𝑒𝑛*1), (*𝑆𝑏*2,*𝑃𝑡𝑦𝑝𝑒𝑛*2), (*𝑆𝑏𝑝*, *𝑃𝑡𝑦𝑝𝑒𝑛𝑝*))  1: **for all** *𝑖* ∈ {1*,* … *, 𝑛*} **do**  2: **for all** *𝑘* ∈ {1*,* … *, 𝑝*} **do**  3: SELECT *𝑈𝑖𝑆𝑏𝑘*; select security behavior trace for user i  4: **for all** *𝑗* ∈ {1*,* … *, 𝑚*} **do**  5: SELECT *𝑆𝑃 𝑟𝑜𝑝𝑗*; select security property to verify  6: PERFORM MODEL CHECKING ON (*𝑈𝑖𝑆𝑏𝑘*) WITH TCTL QUERY  (*𝑆𝑃 𝑟𝑜𝑝𝑗*)  7: STORE SATISFACTION OUTPUT FOR *𝑈𝑠𝑒𝑟𝑖*  in  a list *𝑆𝑎𝑡*\_*𝑂𝑢𝑡𝑖*  = {(*𝑆𝑏𝑘, 𝐵𝑒ℎ*\_*𝐿𝑎𝑏𝑒𝑙𝑘*)}; where  *𝐵𝑒ℎ*\_*𝐿𝑎𝑏𝑒𝑙𝑘*  *𝑖𝑠*  *𝐺𝑘*(*𝐺𝑜𝑜𝑑*) *𝑜𝑟*  *𝐵𝑘*  (*𝐵𝑎𝑑*) *𝑜𝑟*  *𝐺𝐵𝑘*  *𝑒𝑞𝑢𝑎𝑙*  *𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑔𝑜𝑜𝑑 𝑎𝑛𝑑 𝑏𝑎𝑑 𝑠𝑒𝑐𝑢𝑟𝑖𝑡𝑦 𝑏𝑒ℎ𝑎𝑣𝑖𝑜𝑟*  8: **end for**  9: **end for**  10: **end for**  11: **for all** *𝑖* ∈ {1*,* … *, 𝑛*} **do**  12: **for all** *𝑘* ∈ {1*,* … *, 𝑝*} **do**  13: From *𝑆𝑎𝑡*\_*𝑂𝑢𝑡𝑖* Extract behavior *𝑆𝑏𝑘* and *𝐵𝑒ℎ*\_*𝐿𝑎𝑏𝑒𝑙𝑘*  14: **if** {*𝑆𝑏𝑘* ∶ *𝐵𝑒ℎ*\_*𝐿𝑎𝑏𝑒𝑙𝑘* == *𝐺𝑘*} **then**  15:  16: **else if** {*𝑆𝑏𝑘* ∶ *𝐵𝑒ℎ*\_*𝐿𝑎𝑏𝑒𝑙𝑖* == *𝐵𝑘*} **then**  *𝑃𝑡𝑦𝑝𝑒𝑖𝑘* = *𝑟𝑖* ; Least restrictive policy  17: *𝑃𝑡𝑦𝑝𝑒𝑖𝑘* = *𝑠𝑖* ; Strict security policy  18: **else**  19: *𝑃𝑡𝑦𝑝𝑒𝑖𝑘* = *𝑚𝑖* ; Moderate security policy  20: **end if**  21: STORE policy *𝑃 𝑜𝑙𝑖𝑐𝑦𝑖* = {*𝑃𝑡𝑦𝑝𝑒𝑖𝑘*}  22: **end for**  23: PRINT *𝑃 𝑜𝑙𝑖𝑐𝑦𝑖*={*𝑃𝑡𝑦𝑝𝑒𝑖*1, ..., *𝑃𝑡𝑦𝑝𝑒𝑖𝑘*, ..., *𝑃𝑡𝑦𝑝𝑒𝑖𝑝*}  24: **end for** |
| So the policy can be read as follows: For Tom the policy engine should implement least restrictive policy (r) wherever, device securement and 1. *𝑃 𝑜𝑙𝑖𝑐𝑦𝑇 𝑜𝑚* = {DS(G), PG(G), PA(B), U(B)} = ⟨ *𝑟𝑇 𝑜𝑚*, *𝑟𝑇 𝑜𝑚*, *𝑠𝑇 𝑜𝑚*, 2. *𝑃 𝑜𝑙𝑖𝑐𝑦𝑆𝑎𝑟𝑎* = {DS(B), PG(B), PA(G/B), U(B)} = ⟨ *𝑠𝑆𝑎𝑟𝑎*, *𝑠𝑆𝑎𝑟𝑎*, 3. *𝑃 𝑜𝑙𝑖𝑐𝑦𝑍𝑜𝑒* = {DS(G), PG(G/B), PA(NA), U(NA)} = ⟨ *𝑟𝑧𝑜𝑒*, *𝑚𝑧𝑜𝑒*, 4. *𝑃 𝑜𝑙𝑖𝑐𝑦𝐵𝑒𝑙𝑙𝑎* = {DS(B), PG(B), PA(G), U(G)} = ⟨ *𝑠𝐵𝑒𝑙𝑙𝑎*, *𝑠𝐵𝑒𝑙𝑙𝑎*, *𝑟𝐵𝑒𝑙𝑙𝑎*, 5. *𝑃 𝑜𝑙𝑖𝑐𝑦𝐽𝑜ℎ𝑛* = {DS(G), PG(G), PA(G), U(G)} = ⟨ *𝑟𝐽𝑜ℎ𝑛*, *𝑟𝐽𝑜ℎ𝑛*, *𝑟𝐽𝑜ℎ𝑛*, 6. *𝑃 𝑜𝑙𝑖𝑐𝑦𝑍𝑎𝑐* = {DS(G), PG(NA), PA(B), U(NA)} = ⟨ *𝑟𝑍𝑎𝑐*, *𝑚𝑍𝑎𝑐*, *𝑠𝑍𝑎𝑐*, *𝑚𝑇 𝑜𝑚* ⟩*𝑚𝑆𝑎𝑟𝑎*, *𝑠𝑆𝑎𝑟𝑎* ⟩*𝑚𝑧𝑜𝑒*, *𝑚𝑧𝑜𝑒* ⟩*𝑟𝐵𝑒𝑙𝑙𝑎* ⟩*𝑟𝐽𝑜ℎ𝑛* ⟩*𝑚𝑍𝑎𝑐* ⟩  password generation are required, as he has shown good behavior in implementing these security behaviors. Whereas, a strict policy (s) needs to be implemented for proactive awareness, such as by sending more alerts or warnings as, Tom has shown weak behavior in spotting security threats. Finally, moderate policy (m) should be implemented in regards to updates, such as when to send alerts or when to implement specific actions that need to be taken if updates are not installed within a timeline. |

levels of abstraction to represent the knowledge about the potential behaviors. This abstracted representation of security behaviors allowed us to create the required knowledge base as finite state automata and it also allows for scalability, as while checking the satisfiability you are checking one level at a time instead of including all the levels at once. Once modeled, it enabled automated reasoning to check the sat-isfiability of good or bad security behaviors exhibited by users. As a result, using our framework user specific security related policies can be generated. As the policy generation was algorithmic the policy generation can be an automated process for zero trust environment to generate policies with changing user behavior.

We verified 90 properties for six test cases that were generated. The properties were executed on a 64 bit Mac, each property proved within 6–18 s. This higher level reasoning can lead to the generation of parameters to monitor for individual users to capture the user specific behaviors.

**8. Conclusion**

We were able to provide a solution that supports the concept of *Zero Trust* by eliminating the trust, that all users act responsibly. Most importantly, we achieved success in answering the research ques-tion ‘‘How to automatically identify users security practices and then generate security policy after observing and analyzing security behav-iors, especially security-related decisions, exhibited by end-users in an environment with *Zero Trust* assumptions?’’ In our approach, Finite-State Automata supported modeling user security behavior. It allowed showing how a user could transition from safe to unsafe state based on making some specific decisions. TCTL language was used to generate linear-time properties, and thus, with reachability analysis we could check the satisfaction of the security behavior. After observing security behavior and analyzing security behavior, the appropriate policy was assigned to address security gaps caused by specific user.

Our approach demonstrated if we can categorize the behavior of users and capture relevant information regarding the behavior, it en-ables automated reasoning to then identify weaknesses in a users behavior to generate specific policies. Future work involves developing surveys to evaluate if there are more selection predictors. Another extension is to apply the method to accessing network based services, based on the analysis discussed in this research.

**CRediT authorship contribution statement**

**Arwa AlQadheeb:** Acquisition of data, Analysis and/or interpre-tation of data, Writing – original draft. **Siddhartha Bhattacharyya:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Revising the manuscript critically for important intellectual content. **Samuel Perl:** Conception and design of study, Writing – original draft, Revising the manuscript critically for important intellectual content.

**Declaration of competing interest**

The authors declare that they have no known competing finan-cial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgment**

All authors approved the version of the manuscript to be published.

**7. Results**  **References**

Using our FMUSPG framework we were able to select relevant user security behaviors from existing literature. We were then able to decompose the security behaviors exhibited by users at different

10

*A. AlQadheeb et al.*  *Array 14 (2022) 100146*

[2] Gratian M, Bandi S, Cukier M, Dykstra J, Ginther A. Correlating human traits and cyber security behavior intentions. Comput Secur 2018;73:345–58. [http: //dx.doi.org/10.1016/j.cose.2017.11.015](http://dx.doi.org/10.1016/j.cose.2017.11.015).

[3] [West R, Mayhorn C, Hardee J, Mend](http://dx.doi.org/10.1016/j.cose.2017.11.015)el J. Social and human elements of information security: Emerging trends and countermeasures. Hershey, PA: IGI Global; 2009, p. 43–60. <http://dx.doi.org/10.4018/978-1-60566-036-3.ch004>. [4] [Security I. Cost of a dat](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb4)[a breach report. Tech. rep., Ponemon Institute; 2019.](http://dx.doi.org/10.4018/978-1-60566-036-3.ch004) [5] [Egelman S, Peer E. Predicting privacy and security attitudes. ACM SIGC](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb4)AS Comput Soc 2015;45(1):22–8. <http://dx.doi.org/10.1145/2738210.2738215>.

[6] Canali D, Bilge L, Balzarotti [D. On the effectiveness of risk prediction b](http://dx.doi.org/10.1145/2738210.2738215)ased on users browsing behavior. In: ASIA CCS ’14 proceedings of the 9th ACM symposium on information, computer and security. New York, NY: ACM; 2014, p. 171–82. <http://dx.doi.org/10.1145/2590296.2590347>.

[7] [Mitnick KD,](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb7) [Simon WL, Wozniak S. The art of deception](http://dx.doi.org/10.1145/2590296.2590347)[: Controlling the human](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb7)  [element of security. Hoboken, NJ: Wiley; 2002.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb7)

[8] [Curzon P, Rukš ˙enas R, Blandford A. An app](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb7)roach to formal verification of human–computer interaction. Form Asp Comput 2007;19(4):513–50. [http://dx. doi.org/10.1007/s00165-007-0035-6](http://dx.doi.org/10.1007/s00165-007-0035-6).

[9] [A D, M. H.](http://dx.doi.org/10.1007/s00165-007-0035-6)

[10] Bolton M, Bass E, Siminiceanu R. Using formal verification to evaluate human-automation interaction: A review. Syst, Man, Cybern: Syst, IEEE Trans 2013;43:488–503. <http://dx.doi.org/10.1109/TSMCA.2012.2210406>.

[11] [Houser A. Mental](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb11) [models for cybersecurity: A formal methods appr](http://dx.doi.org/10.1109/TSMCA.2012.2210406)[oach. 2018.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb11) [12] [Egelman S, Peer E. Scaling the security wall: Developing a security behavio](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb11)r intentions scale (SeBIS). In: CHI ’15 proceedings of the 33rd annual ACM conference on human factors in computing systems. New York, NY: ACM; 2015, p. 2873–82. <http://dx.doi.org/10.1002/andp.19053221004>.

[13] [Chen B, et al.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb13) [A security awareness and protection system fo](http://dx.doi.org/10.1002/andp.19053221004)[r 5G smart healthcare](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb13)  [based on zero-trust architecture. IEEE Internet Things J 2020;403–16.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb13)

[14] [Mandal S, Khan DA, Jain S. Cloud-based zero trust access control p](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb13)[olicy: An approach to support work-from-home driven by COVID-19 pandemic. New Gener Comput 2021;39.3:599–622.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb14)

[15] [Eidle D, et al. Autonomic](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb14) [security for zero trust networks. IEEE 8th Annual](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb15)  [UEMCON; 2017.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb15)

[16] [Dean E, et al. T](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb15)[oward a zero trust architecture implementation in a university](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb16)  [environment, Vol. 6.4. The Cyber Defense Review; 2021, p. 37–48.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb16)

[17] [West R. The psychology of security. Psychol Secur: Why Do Good](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb16) Users Make Bad Decis? 2008;51(4):34–40. <http://dx.doi.org/10.1145/1330311.1330320>.

[18] Herley C. So long, and no th[anks for the externalities: The rational rejec](http://dx.doi.org/10.1145/1330311.1330320)tion of security advice by users. In: NSPW ’09 proceedings of the 2009 workshop on new security paradigms workshop. New York, NY: ACM; 2009, p. 133–44. <http://dx.doi.org/10.1145/1719030.1719050>.

[19] [Halevi T, Memon N, Lewis J, Kumaraguru P](http://dx.doi.org/10.1145/1719030.1719050), Arora S, Dagar N, et al. 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. New York, NY: ACM; 2016, p. 318–24. [http://dx.doi.org/10.1145/ 3011141.3011165](http://dx.doi.org/10.1145/3011141.3011165).

[20] [Baier C, Katoen JP. Principles of model checking. Cambridge, MA: MIT Press;](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb20)  [2008.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb20)

[21] [Security I. Cost of a data breach report. Tech. rep., Ponemon Institute; 2019.](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb21) [22] [P. CP. RTL hardware design using VHDL: Coding for efficiency, portability, a](http://refhub.elsevier.com/S2590-0056(22)00014-5/sb21)nd scalability. Hoboken, NJ: Wiley; 2006, p. 313–71. [http://dx.doi.org/10.1002/](http://dx.doi.org/10.1002/0471786411.ch10)  [0471786411.ch10](http://dx.doi.org/10.1002/0471786411.ch10).

[23] Cimatti A, Clarke E, E. G, F. G, M. P, M. R, et al. NuSMV 2: An OpenSource tool for symbolic model checking. In: CAV ’02 Proceedings of the 14th international conference on computer aided verification. Berlin, Heidelberg: Springer; 2002, p. 359–64. <http://dx.doi.org/10.1007/3-540-45657-0_29>.

[24] Uppaal. Up[paal website. 2010,](http://dx.doi.org/10.1007/3-540-45657-0_29) <http://www.uppaal.org>[.](http://dx.doi.org/10.1007/3-540-45657-0_29)

[25] Owre S, Rajan S, Rushby JM, Sh[ankar N, Srivas M. PVS](http://www.uppaal.org): Combining specification, proof checking, and model checking. In: 1996 Proceedings of computer aided verification: 8th international conference. Berlin, Heidelberg: Springer; 1996, p. 411–4. <http://dx.doi.org/10.1007/3-540-61474-5_91>.

11