Isabelle Modelchecking for Insider Threats

Florian Kammüller[1]

Middlesex University, London, UK,
f.kammueller@mdx.ac.uk

Abstract. The Isabelle Insider framework formalises the technique of social explanation for modeling and analysing Insider threats in infrastructures including physical and logical aspects. However, the abstract Isabelle models need some refinement to provide sufficient detail to explore attacks constructively and understand how the attacker proceeds. The introduction of mutable states into the model leads us to use the concepts of Modelchecking within Isabelle. Isabelle can simply accommodate classical CTL type Modelchecking. We integrate CTL Modelchecking into the Isabelle Insider framework. A running example of an IoT attack on privacy motivates the method throughout and illustrates how the enhanced framework fully supports realistic modeling and analysis of IoT Insiders.

1 Introduction and Overview

Insider threats pose a serious problem for security and privacy that is inherently hard to control since the attack comes from a user within the security perimeter. Techniques to tackle these challenges need to be application oriented and yet thorough. We propose in this paper to further explore the rigorous modeling and analysis of Insider attacks including infrastructures based on logics: we extend the Isabelle Insider framework with Modelchecking thus combining the practical advantages of policy invalidation with mathematical proof and expressive models. We validate our extension on a case study of a privacy attack on the IoT performed by an Insider.

The original invalidation idea [18] uses the advantages of Modelchecking to find attacks by Insiders on infrastructures. Starting from an invalidated policy, the attempt to modelcheck fails producing an attack vector. The Isabelle Insider framework [20] implements the process of social explanation inspired by Max Weber's work on sociology to model and analyse Insider threats. This framework is a general tool to integrate logical rigour and automated reasoning at the infrastructure level validated on major Insider patterns identified by the CERT Insider guide. Preliminary applications of the Isabelle Insider framework, for example, to Insider threats in the IoT [16] or to Insider threats in aviation [15], show that the framework is capable of expressing realistic case studies. However, to support a systematic identification of detailed attacks from known general attack cases or patterns, more details from the context of the application, e.g. the physical infrastructure, need to be integrated into the model.

The integration of attack trees into the Isabelle Insider framework [14,24] is an important first step to exemplify attack vectors. Attack trees refine an attack case into more detailed sub-attacks which are then successors in the attack tree but they do not provide clues how to find these refinements. For a given attack vector, however, we need a more systematic way to explore the infrastructure graph with its associated actors, policies and credentials, to constructively identify the Insider attack.

In this paper, we revisit the invalidation approach by providing a substantial extension of the Isabelle Insider framework to accommodate Modelchecking of infrastructures. Our contribution is an extension of the Insider framework by a notion of graph-based state and state-transition. In addition, we embed Kripke-structures and the temporal logic CTL into Isabelle to analyse Insider attacks by Modelchecking. Thereby, we finally provide the missing link of the Isabelle Insider framework to the invalidation approach. The extended framework is motivated by and illustrated on an IoT Insider attack.

2 Background

In formal analysis of technical scenarios, the motivation of actors and the resulting behaviour of humans is often not considered because the complexity is beyond usual formalisms. The Isabelle Insider framework [20] provides expressiveness to model infrastructures, policies, and humans while keeping up the level of proof automation. In this section, we give a short summary of this framework for modeling and analysing Insider attacks. A detailed technical introduction is given in [20] and the sources are online [1]. We also present the IoT case study in the current section.

2.1 Social Explanation in Isabelle

The Isabelle Insider framework [20] is based on a logical process of sociological explanation [12] inspired by Weber's *Grundmodell*, depicted in Figure 1, to explain Insider threats by moving between societal level (macro) and individual actor level (micro).

The standard example to illustrate the process of macro-micro-macro transitions in the spirit of Max Weber is to explain the relationship between 'protestant ethic' and 'the spirit of capitalism'. Protestantism has lead to changes in familial socialization, a 'familial revolution' (macro to micro-level). The change of educational style employed by protestant parents (micro-level) has equipped their children with 'strong internalized achievement drives'. This has created the spirit of capitalism back on the collective, the macro-level, and has lead to the spread of a new type of actor, the entrepreneur.

In the application of the steps (a-c) of the logic of explanation, the insider's move over the 'tipping' point is seen as (a), the actual Insider attack as step (b) and the damages caused by the attack as step (c) in Figure 1.

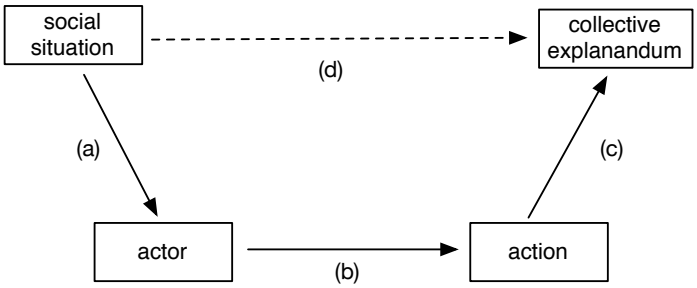

Fig. 1. The 'Grundmodell' of sociological explanation [10]: a macro-micro-macro-transition explains sociological phenomena by breaking down the global facts from the macro level (a) onto a more refined local view of individual actors at the micro-level (b). Finally those micro-steps are generalized and lifted back onto the macro-level (c) to explain the global phenomenon.

The interpretation into a logic of explanation is formalized in Isabelle's Higher Order Logic. This Isabelle formalisation constitutes a tool for proving security properties using the assistance of the semi-automated theorem prover [20]. Isabelle/HOL is an interactive proof assistant based on Higher Order Logic (HOL). Applications can be specified as so-called object-logics in HOL providing reasoning capabilities for examples but also for the analysis of the meta-theory. Examples reach from pure mathematics [17] to software engineering [13]. An object-logic contains new types, constants and definitions. These items reside in a theory file, *e.g.*, the file `Insider.thy` contains the object-logic for social explanation of Insider threats (see [20,1]). This Isabelle Insider framework is a *conservative extension* of HOL. This means that our object logic does not introduce new axioms and hence guarantees consistency.

The micro-level and macro-level of the sociological explanation give rise to a two-layered model in Isabelle, reflecting first the psychological disposition and motivation of actors and second the graph of the infrastructure where nodes are locations with actors associated to them. Security policies can be defined over the agents, their properties, and the infrastructure graph; properties can be proved mechanically with Isabelle. We demonstrate the application of the Isabelle Insider framework in Section 3.1 on an IoT Insider case study presented next.

2.2 Challenge IoT Insider

The Internet-of-Things (IoT) denotes the combination of physical objects with a virtual representation in the Internet. It consists not only of humans but a variety of "Things" as well. From a security and privacy perspective, at this point the IoT could be perceived as a hopeless case since all prevention aspects of security

(confidentiality, integrity, and availability) are inherently weak, and unwanted tracking and monitoring throws the doors wide open to privacy attacks. Insiders using the IoT represent a significant challenge for enterprises. The paper [23] assesses this problem in detail, and outlines several vectors through which insiders may attack their employers. The structure of [23] draws on the VERIS 4A approach to define cyber attacks [27]. This includes understanding the *assets* at risk in the attack, the *actors* (or insiders) that launch the attack, the *attributes* (or impact) of the attack on the asset, and the specific *actions* involved in the attack. Below, we present two of the attack vectors (AVs) from [23] in the broad context of the VERIS approach; the first one perpetrated by a malicious insider (MI) and the second by an unintentional insider (UI) threat. They give a gist of Insider attack cases and the informality of their description. The former one MI-A4 is used as a basis for modeling and analysis throughout this paper in the form of the "Employee Blackmail" case as presented below.

MI-AV4: Using the storage system on a smart device, the insider is able to copy sensitive data (*e.g.*, IP or files) from the organisation's computers to the device and remove it from the enterprise. Bluetooth or NFC may be preferred for this attack as organisations now tend to monitor USB connections. This attack is possible with any IoT device with a storage capability.

UI-AV7: As a result of improperly configured or inadequately protected insider smart devices (*e.g.*, a smart-watch and a paired smartphone), the communications channel between them is compromised by a malicious third-party. This party then gathers enterprise data via the notifications, schedules, messages synchronised across devices. Further detail on such attacks on wearables can be found in [25]. We note that this attack could be conducted by another insider as well. This attack is possible with any device with a notification and storage capability.

2.3 Example – Employee Blackmail

The insider in this case is an employee in the IT department of a manufacturing company. He has received a formal warning from the CEO because there had been reports that the employee had abused colleagues. This warning has been contrived by the CEO himself who had an extramarital liaison with one of the employees with whom the insider had been flirting with. Following that, the IT employee heard rumours that he might be dismissed, which constituted the precipitating event that made him an insider: he planned his revenge.

From a report by an online security blog, the Bitdefender Research Team [4], the insider knew that it was possible to eavesdrop on and intercept communications between a smart-watch and a smartphone. The vulnerability was described in some detail on the blog. So, when the CEO purchased a smart-watch paired with his smartphone, the insider then exploited the vulnerability using additional methods found on hacking forums. He could move freely in the offices and could thus get into close range to collect data communicated between the CEO's

smartphone and smart-watch. Although the communicated data has been encrypted before being transmitted via the Bluetooth protocol, the encryption used a 6-digit PIN code as a key in addition to data obfuscation (adding redundant "padding" to the clear text). Using publicly available decryption algorithms, the insider was thus able to get the key information.

Once the encryption was broken, the Insider could use this credential to collect data on incoming phone calls, SMS and emails, and personal and work related calendar. Finally, the insider blackmailed the CEO with the stolen information that also implied the CEO's liaison with a colleague: he threatened to show it to his wife and children unless he would receive a large severance package and good references. The 4As for this case are as follows:

- *Assets*: Sensitive company and personal information;
- *Actors*: Malicious insider;
- *Attributes*: Unauthorised data access then used for blackmail and fraud; and
- *Actions*: Attack Vector UI-AV7 (where an insider is the perpetrator).

This case highlights a key weakness in IoT devices, i.e., the limited security features with these devices and a clever attack building on personal knowledge helped by current reports and malicious Web forums.

3 IoT Case Study in Isabelle Insider Framework

3.1 Infrastructure Graph and Policies

We now present the formalisation of the 'Employee blackmail' in the Isabelle Insider framework. Isabelle sources of this case study are available [1]. For the application to the office scenario, we only model two identities, `Boss` and `Employee` representing an employee and his boss. The actors that are legal participants of the scenario are summarized in the following set of office actors as a locale definition `office_actors`. The full Isabelle/HOL syntax for a locale definition uses `fixes` and `defines` sections but in all subsequent definitions we omit these and also drop the types for conciseness of the exposition. The double quotes `''s''` create a string in Isabelle/HOL;

```
fixes office_actors :: identity set
defines office_actors_def: office_actors ≡ {''Boss''}
```

In a similarly simplified abstraction, we consider the office's architecture as a simple graph having three locations: employee's office, boss office, and smartphone defined as locale definitions and summarized in the set `office_locations`.

```
sphone ≡ Location 2
boss0 ≡ Location 1
employee0 ≡ Location 0
office_locations ≡ { employee0 , boss0, sphone }
```

As the topology of the infrastructure, we define the following graph where the actors Boss and Employee reside in their respective offices. A graph is quite naturally given as a set of nodes of locations and the actors residing at certain locations are specified by a function associating lists of nodes with the locations.

```
ex_graph ≡  Lgraph {(bossO, employeeO), (employeeO, sphone)}
            (λ x. if x = bossO then [''Boss'']
                  else (if x = employeeO then [''Employee''] else []))
```

In an infrastructure, the actors can have credentials like PINs or they can have roles. We define the assignment of the credentials as predicate over actors. These predicates are true for actors that have these credentials. For the office scenario, the credentials express that the office actor `Boss` possess the PIN for the encryption to the smartphone.

```
ex_creds ≡ (λ x. if x = Actor ''Boss'' then has (x,''PIN'') else True)
```

Similarly, the locations can have features attached to them, like locks. The possible states of the smart phone are *encrypted* or *cracked*. The Isabelle Insider framework provides an additional predicate `isin` that checks the value of a location against string values, here the location `sphone` against the string values `''encrypted''` or `''cracked''`. The following `ex_locks` defines the smart phone to be encrypted.

```
ex_locs ≡ (λ x. if x = sphone then (isin x ''encrypted'') else True))
```

Changing the position of the sphone to `cracked`, i.e., using the PIN of the phone to decypher messages, corresponds to being able to perform a `put` action in the boss's office. The global policy is thus 'no one except the boss can put anything in the boss's office':

```
global_policy I a ≡  a ∉ office_actors ⟶
                     ¬(enables I bossO (Actor a) put)
```

To guarantee this global policy, local policies need to be defined accordingly. These local policies are attached to locations in the organization's graph using a function that maps each location to the set of the policies valid in this location. The policies are again pairs: the first element of these pairs are predicates over actors specifying necessary conditions on actors; the second elements are sets of actions that are authorized in this location for actors authenticated by the predicates. In the following definition of local policies for each node in the office scenario, we additionally include parameters `G`, `ts` and `ls` to refer to the graph, the actors' credentials, and the locations' features. The predicate $@_G$ checks whether an actor is at a given location in the graph G.

```
local_policies G ts ls ≡
 (λ x.  if x = employeeO then {(λ y. True,{get, put, move}) }
        else (if x = bossO then {((λ y. has (y, ''PIN'')), { put}),
                                 (λ y. True, {move})}
              else (if x = sphone then
```

```
{((λ y. (∃ n. (n @_G boss0) ∧ Actor n = x) ∧
 ls sphone = isin sphone ''cracked'',  {get, put}))}
else {})))
```

This policy expresses that any actor can move to the employee's and the boss's office but places the following restrictions on the boss's one.

`put`: to perform a `put` action, i.e., put the PIN and thus crack the phone, an actor must have the PIN;
`move`: to perform a `get` or `put` action at location sphone, i.e., intercept its messages, an actor must be at the position `boss0`, be at position `boss0`, i.e., in the boss's office, and `sphone` must be in state `cracked`.

Although this policy abstracts from the smart watch, and a few other technical details, it contains the essential features of the 'Employee blackmail' scenario. The smart watch and the communication between the watch and the phone is seen in this abstraction as part of the smart phone. The main reason for this coarse abstraction is the conciseness of the exposition in this paper and the fact that PIN and encryption are shared parameters of the combined smart phone-watch system.

The graph, credentials, and features are plugged together with the policy into the infrastructure `Office_scenario`.

```
Office_scenario ≡ Infrastructure ex_graph
                  (local_policies ex_graph ex_creds ex_locs)
                  ex_creds ex_locs
```

3.2 Analysis of Security and Privacy Properties

Note, that all the above definitions have been implemented as local definitions using the locale keywords `fixes` and `defines` [21]. Thus they are accessible whenever the locales `scenarioOffice` is invoked but are not axioms that could endanger consistency. We now also make use of the possibility of locales to define local assumptions. This is very suitable in this context since we want to emphasize that the following formulas are not general facts or axiomatic rules but are assumptions we make in order to explore the validity of the infrastructure's global policy. The first assumption provides that the precipitating event has occurred which leads to the second assumption that provides that Employee can act as an insider:

```
assumes Employee_precipitating_event: tipping_point(astate ''Employee'')
assumes Insider_Employee: Insider ''Employee'' {''Boss''}
```

The above definitions and assumptions provide the model for the Employee blackmail Insider attack. We can now state theorems about the security of the model and interactively prove them in our Isabelle/HOL framework. We first prove a sanity check on the model by validating the infrastructure for the "normal" case. For the boss as an office actor, everything is fine: the global

policy does hold. The following is an Isabelle/HOL theorem `ex_inv` that can be proved automatically followed by the proof script of its interactive proof. The proof is achieved by locally unfolding the definitions of the scenario, *e.g.*, `Office_scenario_def` and applying the simplifier:

```
lemma ex_inv: global_policy Office_scenario ''Boss''
by (simp add: Office_scenario_def global_policy_def office_actors_def)
```

However, since the `Employee` is at tipping point, he will ignore the global policy. This insider threat can now be formalised as an invalidation of the global company policy for `''Employee''` in the following "attack" theorem named `ex_inv1`:

```
theorem ex_inv1:  ¬ global_policy Office_scenario ''Employee''
```

The proof of this theorem consists of a few simple steps largely supported by automated tactics. Thus `Employee` can get access to the data and blackmail the boss. The attack is proved above as an Isabelle/HOL theorem. Applying logical analysis, we thus exhibit that under the given assumptions the organisation's model is vulnerable to an insider.

This analysis follows closely the analysis of Insider attack patterns, like the Entitled independent [20], and applications to Airplane safety and security. The formalization and proofs are very similar. This overall procedure uses the strong assumption that the employee can impersonate the boss thus being able to provide all credentials and act like the boss. The attack stays abstract without explaining in detail how the employee finds the means to get hold of the PIN that then enables him to crack the smart phone. For a more refined approach we would like to be able to demonstrate how it is possible that the employee finds the PIN and breaks the encoding of the smart phone communication with the smart watch.

4 Extensions of Sociological Explanation to State Change

The original approach of invalidation of a global policy based on local policies of infrastructure scenarios [18] uses the idea of Modelchecking: the attempt to prove a security property fails but provides a trace of steps in the infrastructure leading to a state in which the property is violated but more importantly providing a refined attack trace providing detailed steps leading to the attack.

4.1 Refined Attack Scenario

The scenario representing the office in danger, has a graph in which the actor Employee is in the boss's office rather than his own.

```
ex_graph' ≡  Lgraph {(boss0, employee0), (employee0, sphone)}
             (λ x. if x = boss0 then [''Boss'', ''Employee'']
                   else (if x = employee0 then [] else []))
```

The credentials of the actors encode now that the employee has the PIN.

```
ex_creds' ≡ (λ x. if x = Actor ''Boss'' then has (x,''PIN'')
                 else (if x = Actor ''Employee'' then has (x,''PIN'')
                       else True))
```

The location features' settings now encode that the smart phone is cracked.

```
ex_locs' ≡ (λ x. if x = sphone then (isin x ''cracked'') else True))
```

The local policies stay the same as before but we use the updated graph and location settings when re-defining the scenario.

```
Office_in_danger ≡ Infrastructure ex_graph'
                   (local_policies ex_graph' ex_creds' ex_locs')
                   ex_creds' ex_locs'
```

Analysing this new scenario, we can prove that – as before – the insider attack by employee is possible, i.e., the global policy does not hold.

```
¬ global_policy Office_in_danger ''Employee''
```

Note, however, that in this changed infrastructure, the proof is possible without invoking the Insider assumption for Employee. In fact, in this changed infrastructure the employee has already managed – using his privileges as an insider – to get hold of the necessary credentials and use them to manipulate the smart devices and their communication.

The extension that we propose here is to define a notion of state transition between those different states `office_scenario` and `office_in_danger` represented by the respective infrastructure graphs. In introducing the extension to state transition, we will get the benefits of the invalidation approach to Insider attacks that lies in discovering attacks by changing the models – here the infrastructures.

4.2 Infrastructure Graph State Transition

At this point, we have seen that the Isabelle Insider framework allows to model and analyse the IoT scenario by using the standard methodology. However, we have also seen that a detailed analysis of the existing and the changed policies necessitates to change to scenario `Office_in_danger`. This is a scenario that we have extracted from an actual insider attack. How can we ensure that there are no other scenarios that would invalidate the new policy?

The approach taken in the Isabelle Insider framework explores the possible behaviours of actors by a logical exploration of the enables predicate. This exploration starts from one specific infrastructure. As we have seen in this case study, we can model different scenarios by adapting the infrastructure. In the remainder of this section, we want to sketch an extension of the Isabelle Insider framework that generalizes this approach.

We introduce a relation on infrastructures as an inductive predicate called `state_transition` and introduce the syntactic infix notation $I \rightarrow_i I'$ to denote that infrastructures `I` and `I'` are in this relation.

```
inductive state_transition ::
          [infrastructure, infrastructure] ⇒ bool ("_  →ᵢ _")
```

The definition of this inductive relation is given by a set of rules. To give an impression of this definition, we show here just the rule for the move action.

```
move: ⟦ G = graphI I; a @_G l; l ∈ nodes G; l' ∈ nodes G;
        a ∈ actors_graph(graphI I); enables I l (Actor a) move;
        I' = Infrastructure (move_graph_a a l l'
                             (graphI I))(delta I)(tspace I)(lspace I)
      ⟧ ⟹ I →ᵢ I'
```

The rule for `get` allows an actor `a'` to 'nick' something from another actor `a` that is in the same location `l`; in `I'` actor `a'` "has" `z`.

```
get : ⟦ G = graphI I; a @_G l; has (Actor a, z);
        a' @_G l; enables I l (Actor a) get;
        I' = Infrastructure (graphI I)(delta I)
            (λ x. if x = Actor a' then (has (Actor a', z))
                  else (tspace I x))(lspace I)
      ⟧ ⟹ I →ᵢ I'
```

The rule for `put` allows an actor who is in a location and for whom the `put` action is enabled to change the state of that location encoded in the `isin` predicate for some specific location feature of an application.

```
put : ⟦ G = graphI I; a @_G l; enables I l (Actor a) put;
       I' = Infrastructure (graphI I)(delta I)(tspace I)
            (λ x. if x = l then (isin l z) else (lspace I x))
     ⟧ ⟹ I →ᵢ I'
```

We show next how Modelchecking in Isabelle can be constructed over this state transition of the state graph of an infrastructure.

4.3 Modelchecking for Insider Attacks in Isabelle

A very nice and practical feature of Modelchecking is that if the proof of a property fails, a counterexample can be provided automatically. This counterexample consists of a series of steps from the transition relation from an initial state to a state in which the CTL property is violated. In security applications, for example security protocol verification, these sequences of steps correspond to attack sequences. This advantage also goes for Insider attacks and has been exploited in the invalidation approach [18].

Modelchecking is in practice very successful mainly due to its full automation. It is often advertised as a 'push-button' technique in contrast to automated verification techniques, for example with Isabelle, where the user has to interact with the tool to verify properties (although, for example, the applications in the Isabelle Insider framework are mostly performed by quite standard sequences of automatic proof procedures). A major problem of Modelchecking is

the exponential growth of the number of states – and the notorious 'state explosion' problem that arises as a consequence as soon as infinite data domains are considered (which is very common in almost all applications). The practical success of Modelchecking is despite these limitation due to an extension called Symbolic Modelchecking (SMC) which consists in two main technical advances. First SMC represent the next step relation not by explicit states but only by 'symbolic' states that use variables, e.g. x, y. The next step transition relation R then needs to be expressed by specification in terms of these variables using x', y' to denote the successor state of variable x, y. The second technical advance of SMC is the effective representation of boolean formulas over state variables x, y, x', y' in the so called (Ordered) Binary Decision Diagram representation ((O)BDD). (O)BDD are directed acyclic graphs that allow a concise representation for any boolean formula.

Due to the expressiveness of HOL, Isabelle allows us to formalise within HOL the notion of Kripke structures, temporal logic, and formalise the semantics of Modelchecking by directly encoding the fixpoint definitions for each of the CTL operators.

Our encoding of Modelchecking is available online [1]. This encoding is not meant as a competitor for Modelchecking in general but tries to use its good concepts for the analysis of Insider threats. Therefore, the representation of (O)BDD is not important for us here. (O)BDD serves SMC for the efficiency of representation of formulas and we are not concerned about this since Isabelle's Higher Order Logic provides for formula representation. Also, we do not attempt to provide Symbolic Model Checking because to a large extent our representation is largely symbolic since we can use the powerful symbolic language of Higher Order Logic. In addition, we are in fact very specifically interested in some concrete states of certain variables like `has (x, ''PIN'')` for the specification of critical states of infrastructures. Using Isabelle with an embedding of Modelchecking, will on the other hand provide us with the means to explore Kripke structures. The Kripke structures are defined by taking different states of an infrastructure as its states and the state transition relation $\rightarrow_i$ as transition relation R. Via explicit evaluation of temporal logic formulas, the possible paths in the Kripke structure may reveal attacks. Also, it is important to note that the definitions are constructive thus allowing to use Isabelle's code generation technique. Hence, we could actually derive an executable Modelchecker. Another interesting observation is that we can formalise the classic semantic definitions of CTL-operators one-to-one in the context of Higher Order Logic although these operators are defined for propositional logic only. To verify the correctness of our approach, we use some simple implications of essential theorems of Tarski's theory of fixpoints [26] which we can prove in Isabelle (formalisation is available online [1]).

Based on the state transition $\rightarrow_i$ over the infrastructures, we define the CTL-operators `EX` and `AX` expressing that property f holds in some or all next states, respectively.

`AX` $f \equiv$ `{ s. {f0. s` $\rightarrow_i$ `f0 } ⊆` f `}`
`EX` $f \equiv$ `{ s. ∃ f0 ∈` f`. s` $\rightarrow_i$ `f0 }`

The CTL formula `AG` f means that on all paths branching from a state s the formula f is always true (`G` stands for 'globally'). It can be defined using the Tarski fixpoint theory (see the formalisation available online [1]) by applying the greatest fixpoint operator.

```
AG f ≡ gfp(λ Z. f ∩ AX Z)
```

In a similar way, the other CTL operators are defined. Finally, the labeling function for states can be defined for the Isabelle Insider framework as a predicate over infrastructures `I` based on the behaviour definition `enables`.

```
L I ≡ ∃ a l c. enables I a l c
```

Modelchecking a formula f in a Kripke structure `M` for Insiders can now be defined formally in Isabelle by stating that the initial states of the Kripke structure `init M` need to be contained in the set `L s` of all states `states M` that imply f.

```
M ⊢ f ≡  init M ⊆ { s ∈ (states M) . s ∈ f ∧ (L s) }
```

The set of states of the Kripke structure can be defined as the set of states reachable by the infrastructure state transition from the initial state `Office_scenario`.

```
Office_states ≡ { I. Office_scenario →ᵢ^*  I }
```

The relation `→ᵢ^*` is the reflexive transitive closure – an operator supplied by the Isabelle theory library – applied to the relation $\rightarrow_i$.

The `Kripke` constructor combines the constituents initial state, state set, state transition relation $\rightarrow_i$ and labeling function `L`.

```
Office_Kripke ≡ Kripke Office_states {Office_scenario} →ᵢ L
```

When we now try to verify the global security policy, the attempt to prove fails.

```
Office_Kripke ⊢ AG global_policy
```

In order to explore more precisely where it fails, we prove the complementary property.

```
Office_Kripke ⊢ EF ¬ global_policy
```

This final proof reveals the chain of actions that leads to the attack state `Office_in_danger`: from the initial state `Office_scenario` a state transition (by rule `move`) moves the boss to the employee's office. Then, employee can get the `PIN` – corresponding to a transition with rule `get`. Finally, the employee can move to the boss's office and is in the possession of the `PIN` leading to the infrastructure `Office_in_danger`. Besides showing the trace of actions, this chain of actions also highlights the different state graphs traversed by the state transition relation $\rightarrow_i$ on the way from the initial `Office_scenario Office_in_danger`. The integration of the contextual information into the graphical model permits the systematic exploration of the actions' effects on the infrastructure leading to the attack state. Summarizing, the Modelchecking approach to invalidation can be integrated into the Isabelle Insider framework.

5 Related Work and Conclusions

In this paper, we have provided an extension of the Logical Explanation for Insider Threats with the Isabelle Insider framework to Modelchecking. As requirements elicitation we used the case study of an IoT Insider attack. We used the Isabelle Insider framework for the formal modeling showing where abstract models need more refinement in order to provide sufficient detail to document the attack and make it more realistic. The introduction of mutable states into the model has lead us to use the concepts of Modelchecking. We have then shown how Isabelle can be extended to accommodate classical CTL type Modelchecking and how this extension can be smoothly integrated into the Isabelle Insider framework. The case study of the IoT attacker illustrates how the enhanced framework now fully supports realistic modeling and analysis of IoT Insiders.

The Insider threat patterns provided by CERT [6] use the System Dynamics models, which can express dependencies between variables. The System Dynamics approach has also been successfully applied in other approaches to Insider threats, for example, in the modeling of unintentional insider threats [11]. Axelrad et al. [2] have used Bayesian networks for modelling Insider threats in particular the human disposition. In comparison, the model we rely on for modeling the human disposition is the Isabelle Insider framework, a simplified classification following the taxonomy given in [22]. On the other side, compared to all these approaches, the Isabelle Insider framework provides an additional model of infrastructures and policies allowing reasoning at the individual and organisational level.

On the formal side within the Insider threat community in general, the work by Bishop et al [3] is relevant to the Isabelle Insider framework since it also uses a formal model to analyse Insider threats. Bishop and colleagues use the LITTLE-JIL process description language, a general framework for Software Engineering. It allows the definition of activities, artifacts, and agent specifications. For the analysis, they use fault tree analysis and finite state verification. While resembling the Isabelle Insider framework concepts, in comparison, the Isabelle framework provides more support to express organisations' infrastructures. The ready made analysis procedures of LITTLE-JIL provide an easier to use analysis approach while Isabelle is superior in flexibility, expressivity and thus generality when it comes to properties.

Logical modeling and analysis of Insider threats has started off by investigating Insider threats with invalidation of security policies in connection with Modelchecking [18,19]. This early approach also uses infrastructure models of organisations, actors and policies but necessarily has to be simpler than the Isabelle Insider framework since model checking does only support finite models. The use of sociological explanation has been pioneered in [5] already with first formal experiments in Isabelle. Finally, the Isabelle Insider framework has been established [20] and has been validated on two of the main three Insider patterns the Entitled Independent and Ambitious Leader. Recently an application to IoT Insiders [16] has consolidated the applicability of the Isabelle Insider framework but also illustrated an extension of the framework to attack trees. Attack trees

have been added to the Isabelle Insider framework [24] to provide the possibility to refine attacks once they have been identified. This refinement is formalised together with the notion of attack trees as first introduced in [14]. Another extension towards probabilistic modeling using Bayes networks (BN) and Markov decision processes (MDP) has been explored in [7] but not within the Isabelle Insider framework. Although the work follows the concept of sociological explanation, the tool Matlab is used for the analysis of the micro-level BN and the Prism Modelchecker provides an analysis of the infrastructure's representation as MDP.

Beyond the current state of the Isabelle Insider framework, the application presented in this paper has shown that a more thorough Insider analysis might be achieved by generalising the approach of considering different infrastructures by defining an inductive relation on them. We have intentionally named this relation 'state transition' to refer to the idea of model checking that has initially inspired the logical approach. We have provided an embedding of the concepts of model checking in Isabelle. On top of the induction relation, a notion of validity of formulas in a Kripke structure in combination with temporal logic has been provided in Isabelle. Embedding Modelchecking into Isabelle has been done before, e.g. [9], but not in the context of Insider threat analysis. An interesting observation, however, is that the classical CTL model checking methodology usually restricted to propositional logic can be applied to Higher Order Logic formulas.

References

1. F. Kammüller. Isabelle Insider framework including Modelchecking and Examples, 2016. Available from `https://www.dropbox.com/sh/rx8d09pf31cv8bd/AAALKtaP8HMX642fi04Og4NLa?dl=0`.
2. E. T. Axelrad, P. J. Sticha, O. Brdiczka, and J. Shen. A Bayesian network model for predicting insider threats. IEEE Security and Privacy Workshops, SPW–WRIT, 2013.
3. M. Bishop, H. M. Conboy, H. Phan, B. I. Simidchieva, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, and S. Peisert. Insider threat identification by process analysis. IEEE Security and Privacy Workshops, SPW–WRIT, 2014.
4. Bitdefender. Bitdefender research exposes security risks of android wearable devices, 2014. Avaialable from `http://www.darkreading.com/partner-perspectives/bitdefender/bitdefender-research-exposes-security-risks-of-android-wearable-devices-/a/d-id/1318005`.
5. J. Boender, M. G. Ivanova, F. Kammüller, and G. Primiero. Modeling human behaviour with higher order logic: Insider threats. In *STAST'14*. IEEE, 2014. co-located with CSF'14 in the Vienna Summer of Logic.
6. D. M. Cappelli, A. P. Moore, and R. F. Trzeciak. *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.
7. T. Chen, F. Kammüller, I. Nemli, and C. W. Probst. A probabilistic analysis framework for malicious insider threats. *Human Aspects of Information Security, Privacy, and Trust - Part of HCI International 2015*, *LNCS* **9190**, Springer, 2015.

8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
9. J. Esparza, P. Lammich, R. Neumann, T. Nipkow, A. Schimpf, and J.-G. Smaus. A fully verified executable ltl model checker. *Computer Aided Verification (CAV 2013)*, *LNCS* **8044**, Springer, 2013.
10. H. Esser. *Soziologie – Allgemeine Grundlagen*. Campus, 1993.
11. F. L. Greitzer, J. R. Strozer, S. Cohen, A. P. Moore, D. Mundie, and J. Cowley. Analysis of unintentional insider threats deriving from social engineering exploits. IEEE Security and Privacy Workshops, SPW–WRIT, 2014.
12. C. G. Hempel and P. Oppenheim. Studies in the logic of explanation. *Philosophy of Science*, 15:135–175, April 1948.
13. L. Henrio, F. Kammüller, and M. Rivera. An asynchronous distributed component model and its semantics. In *Formal Methods for Components and Objects*, *LNCS* **5751**, Springer, 2009.
14. M. G. Ivanova, C. W. Probst, R. R. Hansen, and F. Kammüller. Transforming graphical system models into graphical attack models. *Graphical Models for Security, GraMSec'15*, *LNCS*. Springer, 2015. co-located with CSF'15.
15. F. Kammüller and M. Kerber. Investigating airplane safety and security against insider threats using logical modeling. In *IEEE Security and Privacy Workshops, SPW–WRIT*. IEEE, 2016.
16. F. Kammüller, J. R. C. Nurse, and C. W. Probst. Attack tree analysis for insider threats on the iot using isabelle. In *Human Aspects of Information Security, Privacy, and Trust - Part of HCI International 2016 LNCS*. Springer, 2016.
17. F. Kammüller and L. C. Paulson. A formal proof of sylow's theorem. *Journal of Automated Reasoning*, 23(3):235–264, 1999.
18. F. Kammüller and C. W. Probst. Invalidating policies using structural information. IEEE Security and Privacy Workshops, SPW–WRIT, 2013.
19. F. Kammüller and C. W. Probst. Combining generated data models with formal invalidation for insider threat analysis. IEEE Security and Privacy Workshops, SPW–WRIT, 2014.
20. F. Kammüller and C. W. Probst. Modeling and verification of insider threats using logical analysis. *IEEE Systems Journal, Special issue on Insider Threats to Information Security, Digital Espionage, and Counter Intelligence*, 2016.
21. F. Kammüller, M. Wenzel, and L. C. Paulson. Locales - a sectioning concept for isabelle. *Theorem Proving in Higher Order Logics, TPHOLs'99*, *LNCS* **1690**. Springer, 1999.
22. J. R. C. Nurse, O. Buckley, P. A. Legg, M. Goldsmith, S. Creese, G. R. T. Wright, and M. Whitty. Understanding Insider Threat: A Framework for Characterising Attacks. IEEE Security and Privacy Workshops, SPW–WRIT, 2014.
23. J. R. C. Nurse, A. Erola, I. Agrafiotis, M. Goldsmith, and S. Creese. Smart insiders: Exploring the threat from insiders using the internet-of-things. *International Workshop on Secure Internet of Things 2015 (SIoT 2015), in conjunction with ESORICS'15*, *LNCS*. Springer, 2015. In print.
24. C. W. Probst, F. Kammüller, and R. R. Hansen. Formal modelling and analysis of socio-technical systems. *Semantics, Logics, and Calculi, (Nielsens' Festschrift)*. Springer, 2016.
25. Symantec. How safe is your quantified self?, 2014. Tech. Rep.
26. A. Tarski. A lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
27. VERIS. Veris: The vocabulary for event recording and incident sharing, 2015. Available from `http://veriscommunity.net`.