Investigating Airplane Safety and Security against Insider Threats Using Logical Modeling

Florian Kammüller
Department of Computer Science
Middlesex University London
f.kammueller@mdx.ac.uk

Manfred Kerber
School of Computer Science
University of Birmingham
M.Kerber@cs.bham.ac.uk

***Abstract*—In this paper we consider the limits of formal modeling of infrastructures and the application of social explanation for the analysis of insider threats in security and safety critical areas. As an area of study for the analysis we take examples from aviation, firstly since incidents are typically well-documented and secondly since it is an important area per se. In March 2015, a Germanwings flight crashed in the French Alps in what is quite firmly believed to have been intentionally caused by the copilot who locked the pilot out of the cockpit and programmed the autopilot on constant descent. We investigate the security controls and policies in airplanes against insider threats using logical modeling in Isabelle.**

I. Introduction and Overview

In this paper, we take a critical look at logical methods to model and analyze safety and security critical systems. In particular, we address recent advances in applying formal verification techniques, e.g. [1], in the context of severe incidents like the tragic crash of a Germanwings flight: "On March 24, 2015 Germanwings Flight 9525, a scheduled flight from Barcelona to Düsseldorf was hijacked by the co-pilot. 30 minutes after takeoff Andreas Lubitz locked himself in [the] cockpit when [the] captain went out for a rest. Then the co-pilot started to descend. [The] captain ... tried to communicate with Lubitz, but he didn't reply. After 8 minutes of falling the airplane crashed in the Alps near the French village Prads-Haute-Blone. There were 144 passengers and 6 crew members on board. None of them survived the crash." [2]

In this paper, we first summarize the development of airplane security (Section II) before we model and analyse the specific settings in passenger transport airplanes using the Isabelle Insider framework (Section III) leading to a discussion and conclusions (Section IV).

It should be noted that with insider attacks there are two fundamentally different approaches. The first is based on probabilistic reasoning and cost analysis. For instance, if in a company there is a leakage in low-cost pens – since employees take them home – then the company must consider whether imposing a strict control regime involving CCTV cameras is cost effective in eliminating the problem. The company may conclude that cheaper measures minimize the overall cost of lost pens and of measures to lose only few.

The second approach is based on Boolean reasoning in which it is investigated whether certain attacks are possible or not. That is, a model of a real world situation is built and then it is checked or proved that a particular situation can occur or not, for instance, by checking whether a particular state is reachable from an initial state or not. Such an approach would be applied, for instance, if a particular situation is unacceptable and has to be avoided at all cost. As a consequence, for airplane safety this will be the method of choice and is the approach taken in the following. Of course, this does not mean that the approach gives 100% guarantees. The main problem is that any model of a real world situation is necessarily incomplete since it abstracts from (hopefully only irrelevant) details. However, it is difficult to establish that nothing essential has been overlooked.

A. Related Work

We consider here some main threads of work by others while discussing the relationship of the contribution of this paper to our own previous work in Section IV.

The Insider threat patterns provided by CERT [3] use the System Dynamics models, which can express dependencies between variables. The System Dynamics approach is also successfully being applied in other approaches to Insider threats, for example, in the modeling of unintentional insider threats [4]. Axelrad et al. [5] 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 in the Isabelle Insider framework is a simplified classification following the taxonomy provided by [6]. In contrast to all these approaches, our work provides an additional model of infrastructures and policies allowing reasoning at the individual and organisational level.

In the domain of rigorous analysis of airplane systems, work often follows for practical and economic reasons a philosophy of using a mix of formal and systematic informal methods. An example from airplane maintenance procedures [7] uses a security evaluation methodology following the Common Criteria and a formal model and verification with the model checker AVISPA. In comparison, we use a more expressive logical model in the Isabelle Insider framework than their AVISPA specification. Also, we believe that our work is the first to consider Insider threats within airplane safety and security at least in a formal way.

On the formal side within the Insider threat community in general, the work by Bishop et al. [8] 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, expressiveness and thus generality when it comes to properties.

II. Development of Airplane Safety and Security

On 2001-09-11, four terrorist attacks took place in the USA, two on the two towers of the World Trade Center, one on the Pentagon, and in a fourth attack the airplane crashed when passengers tried to overcome the hijackers (a detailed description of the events and a list of aircraft hijackings can be found on Wikipedia [2], [9], including more than 300 further pointers). Before these attacks, aircraft hijacking typically meant that the hijackers had some negotiable demands. Because of the risk to life for the people on board the aircraft, the standard approach was to enter negotiations and to avoid a resolution by force while the aircraft was in the air.

In particular, also there was no secured door between the passenger compartment and the cockpit in airplanes, actually the door was occasionally open, even allowing passengers to get a glimpse of the cockpit during the flight. In Western countries there were no airplane hijackings with major loss of life between the 1970s and the 2001-09-11 attacks. This could be interpreted in the USA and other countries as creating a false sense of security. In the wake of the attacks a serious rethink of the security provision has happened. In particular, the cockpit doors were reinforced and made bullet-proof, making it nearly impossible to open by intruders [10].

These (and other) changes seem to have had the wanted effect, since in the 14 years after the introduction of secured cockpit doors there were only 14 airplane hijackings[1] (as listed on [2]), all but one of them could be prevented from causing fatalities, and the one that did result in fatalities was an insider attack. One nearly successful airplane hijacking has been caused by the copilot who forced Ethiopian Airlines Flight 702 to land at Zurich airport in an attempt to blackmail asylum for himself in Switzerland [2]. Also this airplane hijacking can be characterized as an Insider attack since the attacker was part of the crew.

[1]Note however that there were other attacks on flights which did not originate from passengers, such as the Malaysia Airline Flight MH17 which was brought down by a missile over Ukraine on 2014-07-17.

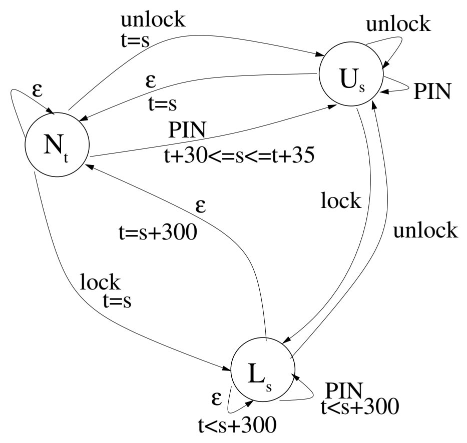

Figure 1. A finite timed automaton to describe the locking mechanism of the door

The one major exception to the rule was Germanwings Flight 9525 on 2015-03-24, which was on the way from Barcelona to Düsseldorf. The aircraft was hijacked by the copilot who locked out the captain who had left the cabin. The pilot tried to regain access to the cockpit but did not succeed. Subsequently, the copilot brought the aircraft to a crash in which all 150 people on board died.

Let us now look more closely into the door and its release mechanism.[2] The door is operated by a switch from inside the cockpit (with three positions: "unlock", "norm", "lock") and a keypad outside the cockpit. In order to gain access to the cockpit normally a crew member would use the interphone to contact a pilot in the cockpit to request access, then presses the hash key on the keypad, which triggers a buzzer in the cockpit, and the pilot releases the door using the switch to open the door (by keeping it in the "unlock" position). In case the pilot(s) is/are incapacitated the crew member outside the cockpit can enter an emergency code to open the door. After 30 seconds (during which the buzzer sounds in the cockpit) of no reaction by the pilots the crew member can open the door for five seconds.

Since this access method could be used by a hijacker to force a crew member to open the door from outside the cockpit, the pilots can, within the 30 seconds between entering the emergency code and the release of the door, lock the cockpit door by putting the toggle button into the "lock" mode. In that case the keypad is disabled for five

[2]The information is extracted from a 5:32 film by Airbus [11].

minutes and the door can be opened during this time only from inside the cockpit by putting the button in the position “unlock”.

The mechanism can be described on different levels and each level requires certain assumptions (e.g., that the door itself will withstand any physical force that may be exerted by an attacker). According to Occam’s Razor, we try to give a representation that is as easy as possible and still describes the situation in sufficient detail that the important aspects are modelled. A first approximation can be given by the timed finite state machine in Figure 1 with three states “U”, “N” and “L” for “unlock”, “norm”, and “lock”, respectively. While time plays a role and it makes a difference for humans whether the door is locked for 300ms, 300s, of 300 minutes, we will abstract from this in the following formalization. During the fatal flight, the copilot used this locking mechanism to lock out the captain from the cockpit. While the mechanism has been successful so far from preventing any fatal attempt by an outsider to hijack an aircraft, the same mechanism prevented the captain from re-entering the cockpit and take action to rescue the aircraft in this case.

In the investigation of the crash, it was found that the copilot suffered from depression and was declared by a doctor unfit to work for the day of the crash. However, he did not forward the sick note to his employer and his doctor was in no position to inform the employer of a potential risk because of medical secrecy. The exact reasons why the copilot killed himself and also why – assumed he was determined to commit suicide – he did not choose a method that did not involve others is unclear and open to some speculation. A contributing factor may have been that the status of his pilot license was conditional. As the investigation found out later [12], a previous episode of medical depression of the copilot not only led to an interruption of his training, but also his license had a lock flag that medical consultation had to be taken before any extension. This required additional scrutiny may have led in him to an increased fear of losing his job and livelihood and may have been a contributing factor to the fatal decision he took.

As a consequence of the events on this day, a number of countries made the rule or recommendation that requires two authorized personnel to be present in the cockpit of large passenger aircrafts at all times [12]. Furthermore there were discussions to loosen the rules of medical secrecy in Germany. The two-person rule was implemented within days by many major airlines worldwide (assumed they did not have it already before the incident) without any discussion of possible negative consequences. The status of loosening medical secrecy is less clear. The argument for it is that if a pilot is a danger then the airline should know that and be able to take action. The counter-argument is that this may lead to the situation that pilots who need help do not dare to seek help any longer since they must fear to lose their jobs.

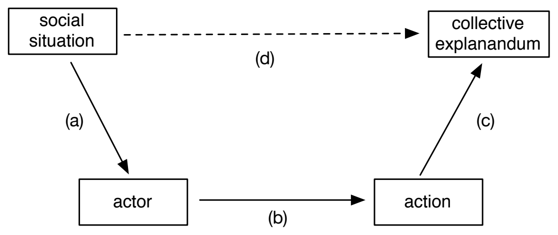

Figure 2. The ‘Grundmodell’ of sociological explanation [15]: a macro-micro-macro-transition explains sociological phenomena by breaking down the global facts from the macro level (a) onto a more refined local view of individual actors at the micro-level (b). Finally those micro-steps are generalized and lifted back onto the macro-level (c) to explain the global phenomenon.

III. Formal Model and Analysis

In formal analysis of technical scenarios, the motivation of actors and the resulting behaviour of humans is often not considered. In this paper, we validate an approach to model and analyse Insider attacks provided by an Insider framework based on the interactive proof assistant Isabelle on the airplane scenario. Isabelle sources are available [13].

A. Social Explanation in Isabelle

The Isabelle Insider framework [1] is based on a logical process of sociological explanation [14] inspired by Weber’s *Grundmodell*, depicted in Figure 2, 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 our application of the steps (a-c) of the logic of explanation, we see the insider’s move over the ’tipping’ point as (a), the actual Insider attack as step (b) and the damages caused by the attack as step (c) in Figure 2.

The interpretation into a logic of explanation is formalized in Isabelle’s Higher Order Logic. This Isabelle formalization constitutes a tool for proving security properties using the assistance of the semi-automated theorem prover [1]. 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 [16] to software engineering [17]. 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 described in the following paragraph. 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.

B. Isabelle Insider Framework

In the Isabelle/HOL framework for Insiders, policies are predicates using the basic actions `get`, `move`, `eval`, and `put` serving for the specification of the capabilities of actors within scenarios. Actions have no parameters which is a fairly strong abstraction but provides generality.

```
datatype action = get | move | eval | put
```

As a representation of human factors we use an abstract type `actor` with elements created by a function `Actor` over identities which are again represented as strings, *e.g.*, `Actor ''Bob''` is the actor named Bob.

```
typedecl actor
type˙synonym identity = string
consts Actor :: string ⇒ actor
```

Policies define the conditions for actions to be permitted to actors. A policy is thus a pair containing a predicate, the selection of the actors, and a set of actions that are permitted to the selected actors.

```
type_synonym policy = ((actor ⇒ bool) × action set)
```

The physical layout of the scenarios is represented as discrete graphs, i.e., sets of pairs of nodes. In addition, actors can reside at nodes within this graph, i.e., nodes have lists of identities assigned to them representing the current location of these actors.

```
datatype location = Location nat
datatype node = Node location "identity list"
datatype 'n graph = Graph ('n × 'n)set
```

We integrate policies and a graph representing the scenario into the infrastructure where policies reside at locations and actors and locations have additional predicates to express specific properties like actors' credentials or state components at locations.

```
datatype infrastructure =
Infrastructure "node graph" "location ⇒ policy set"
               "actor ⇒ bool" "location ⇒ bool"
```

Local policies provide a specification of the behaviour of actors if they remain within the limits of the possible. The following `enables` predicate specifies this concept. In the infrastructure `I`, an actor `a` can perform an action `a'` at location `l` if:

- there is a local policy `(p,e)` stored at location `l` in the infrastructure `I`'s policy accessible as `delta I l`,
- the action `a'` is contained in the set of actions `e`, and
- the policy condition `p` for actor `a` is true under the additional assumption of
  - the actor's `a` credentials `tspace I a` and
  - the location features' settings `lspace I l`.

```
enables I l a a' ≡ ∃ (p,e) ∈ delta I l.
     a' ∈ e ∧ (tspace I a ∧ lspace I l) ⟶ p(a)
```

For the modelling of the micro-level, the individual's disposition, the Isabelle Insider framework relies on a characterization of insider threats [6] that offers a taxonomy of insider threats. This taxonomy is based on a thorough survey on results from counterproductive workplace behaviour, e.g., [18], [19] and case studies from the CMU-CERT Insider Threat Guide [3]. The Insider framework simply models the taxonomy in HOL as datatypes, a concept of HOL that resembles the concept of taxonomy classes. As an example, consider the formal representation of *Psychological State* [6] as a datatype.

```
datatype psy_states = happy | depressed | angry
                    | disgruntled | stressed
```

The element on the right hand side are the five injective constructors of the new datatype `psy_states`. They are simple constants, modeled as functions without arguments. Another example is *Motivation* [6].[3]

```
datatype motivations = financial | political | fun
                     | power | competitive_advantage
                     | revenge | peer_recognition
```

A practical issue is the integration of causalities, quantification or qualification into this basic model. For example, if an employee is disgruntled this might give rise to a motivation of revenge. In [6], these causalities are expressed by drawing lines between boxes containing the classes of the taxonomy. These dependencies resemble the relation between variables in the System Dynamics model. Such lines express dependencies, like 'motivation for revenge may be caused by anger' but this is not a logical causality, i.e., anger ⇒ revenge – a logical causality expresses that anger necessarily implies revenge motivation which might not be the case for all actors.

The *Precipitating Event* or *Catalyst* has a separate role in the characteristics given in the taxonomy. It can be any event that has the potential to tip the insider over the edge into becoming a threat to their employer. It has been called the 'tipping point' in the literature and can be formalized as a predicate on actors. In order to carry over to the micro-level

[3]It is not clear that any of these adequately describes the motivation of the co-pilot of the fatal Germanwings flight. Closest may come `revenge`, however, in the context of possible depression it seems to be at best only partially appropriate.

representation, it is useful to integrate this predicate with the various characteristics about the actor in a combined state.

```
datatype actor_state = State motivation psy_state
```

Finally, the catalyst is encoded as a tipping point predicate that describes the motivation of an actor to become an insider.

```
definition tipping_point :: actor_state ⇒ bool
tipping_point a ≡ motivation a ≠ {}
                  ∧ happy ≠ psy_state a
```

Attacks on security protocols, like the classical Needham-Schroeder public key attack and other insider threat case studies, show that a recurring scheme in insider attacks lies in role identification as described in [20]. The Isabelle Insider framework uses this role identification in the definition of the `UasI` predicate. It expresses that the insider plays a loyal member of an organization while simultaneously acting as an attacker.

```
UasI a b ≡ (Actor a = Actor b)
```

Insider attacks link the micro level insider characterization of psychological disposition with the above insider behaviour `UasI`. This is defined by the following rule `Insider a C` for the attacker `a`. The parameter `C` is a set of identities representing the members of an organisation that are to be considered as safe.

```
Insider a C ≡
tipping_point (astate a) ⟶ (∀ b ∈ C. UasI a b)
```

Although the above Insider predicate is a rule, it is not axiomatized. It is just an Isabelle definition i.e., it serves as an abbreviation. To use it in an application, like the Airplane scenario, we can use this rule as a local assumption (using the `assumes` feature of locales [21]).

C. Airplane Scenario in Isabelle Insider Framework

In the Airplane scenario we use four identities: Bob, Charly, Alice, and Eve. Bob is the pilot, Charly is the copilot, Alice the flight attendant, and Eve is the malicious agent that can act as the copilot. The actors that are legal participants of the scenario are summarized in the following set of airplane actors as a locale definition. The full Isabelle/HOL syntax for a locale definition uses a `fixes` and `defines` keyword but we drop this and the types for conciseness of the exposition in subsequent definitions. The double quotes `''s''` create a string in Isabelle/HOL.

```
fixes airplane_actors :: identity set
defines office_actors_def:
   office_actors ≡ {''Bob'', ''Charly'', ''Alice''}
```

In a similarly simplified abstraction, we consider the airplane's architecture as a simple graph having three locations: `cockpit`, `door`, and `cabin` defined as locale definitions and summarized in the set `airplane_locations`.

```
cockpit ≡ Location 2
door ≡ Location 1
cabin ≡ Location 0
airplane_locations ≡ { cabin, door, cockpit }
```

As the topology of the infrastructure, we define the following graph where the actors Bob and Charly reside in the cockpit and Alice in the cabin.

```
ex_graph ≡  Graph {
 (Node cockpit [Actor ''Bob'', Actor ''Charly''],
  Node door []),
 (Node door [], Node cabin [Actor ''Alice'']) }
```

In an infrastructure, the actors can have credentials like PINs or they can have roles. We define the assignment of the credentials as predicates over actors. These predicates are true for actors that have these credentials. For the Airplane scenario, the credentials express that the airplane actors have the intended roles and all possess the PIN for the door.[4]

```
ex_creds ≡ (λ x.
  (if x = Actor ''Bob''
   then role(x, ''pilot'') ∧ has (x,''PIN'')
   else (if x = Actor ''Charly''
         then role(x,''copilot'') ∧ has (x,''PIN'')
         else (if x = Actor ''Alice''
               then role(x, ''flightattendant'')
                   ∧ has (x,''PIN'')
               else True))))
```

Similarly, the locations can have features attached to them, like locks. To describe the possible locking states of the door we first define a datatype.

```
datatype doorstate = lock | norm | unlock
```

The predicate `ex_locs` is a predicate over locations and uses another predicate `isin_l` that checks the value of a location against values, here against the values of type `doorstate`.

```
ex_locs ≡ (λ x. if x = door then (isin_l x norm)
              else (if x = cockpit
                    then (isin_l x air) else True))
```

Similar to the door, the flight status of the airplane is defined as its position by a datatype which is used in the above `ex_locs` predicate to set the position to "air" by `isin_l cockpit air`.

```
datatype position = air | airport | ground
```

Changing the position of the airplane to ground, i.e., landing it outside airports, like in emergencies or attacks, corresponds to being able to perform a `put` action in the cockpit.

The global policy for the Airplane scenario is thus 'no one except airplane actors can perform `put` actions at location cockpit':

```
global_policy I a ≡  a ∉ airplane_actors ⟶
```

[4]Note that λ in the following is the usual lambda-operator of higher order logic that describes functions. For instance, the square function can be defined – without giving it a name – as $\lambda x. x * x$.

```
                    ¬(enables I cockpit (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 Airplane 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 y. if y = cockpit then
{(λ x. (∃ n. (n @G cockpit) ∧ Actor n = x), {put}),
 (λ x. (∃ n. (n @G door) ∧ Actor n = x ∧
             has (x, ''PIN'')∧
             ls door = isin_l door norm), {move})
}
else (if y = door then {(λ x. True, {move})}
      else (if y = cabin then {(λ x. True, {move})}
            else {}))
```

This policy expresses that any actor can move to door and cabin but places the following restrictions on cockpit.

- `put`: to perform a put action, i.e., put the plane into a new position or put the lock, an actor must be at position cockpit, i.e., in the cockpit;
- `move`: to perform a move action at location cockpit, i.e., move into it, an actor must be at the position door, must be in possession of PIN, and door must be in state norm.

Although this policy abstracts from the buzzer, the 30 sec delay, and a few other technical details, it contains the essential features of the cockpit door.

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

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

D. Analysis of Safety and Security 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 locale `airplane` is invoked. But since definitions are essentially abbreviations, they adhere to the principle of conservative extension of HOL not endangering consistency. However, we make also use of local assumptions within locales. This is part of the reasoning process: the following formulas are not axioms but are locally assumed to analyse the infrastructure's policies. The main assumption is that the precipitating event has lead Eve to tipping point leading to the second assumption that Eve is an Insider impersonating Charly.

```
assumes Eve_precipitating_event:
                         tipping_point(astate ''Eve'')
assumes Insider_Eve : Insider ''Eve'' {''Charly''}
```

The above definitions and assumptions provide the model for the Airplane 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 pilot Bob as an airplane 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.*, `Airplane_scenario_def` and applying the simplifier:

```
lemma ex_inv:
  global_policy Airplane_scenario ''Bob''
by (simp add: Airplane_scenario_def
              global_policy_def airplane_actors_def)
```

We can also show that the same holds for the copilot `Charly`.

```
  global_policy Airplane_scenario ''Charly''
```

However, since `Eve` is an insider who can impersonate Charly, she will ignore the global policy. This insider threat can now be formalized as an invalidation of the global company policy for `''Eve''` in the following "attack" theorem named `ex_inv2`:

```
theorem ex_inv1:
   ¬ global_policy Airplane_scenario ''Eve''
```

The proof of this theorem consists of a few simple steps largely supported by automated tactics. Thus `Eve` can get access to the cockpit and put the position to `ground` – crash the plane. The attack is proved above as an Isabelle/HOL theorem.

This analysis follows closely the analysis of Insider attack patterns, like the Entitled independent [1], and applications to the Internet of Things (IoT) [22]. The formalization and proofs are very similar.

For the current application to the Airplane scenario, the analysis must go beyond the standard steps to provide a better understanding of the risks and possible improvements to the policies and the analysis techniques.

Considering the requirements on Safety and Security that have evolved over the years as sketched in Section II, we can distinguish in the current airplane policies two requirements, one which we describe here as

- *Safety:* if the actors in the cockpit are out of action, there must be a possibility to get into the cockpit from the cabin, and

Security: if the actors in the cockpit fear an attack from the cabin, they can lock the door.

Based on the specification of the Airplane scenario, we can express Safety quite concisely by stating that airplane actors can move into the cockpit.

```
Safety I a ≡ a ∈ airplane_actors
             ⟶ (enables I cockpit (Actor a) move)
```

Security can nearly as simply be defined as the property that no Actor can move into the cockpit if the door is on lock.

```
Security I a ≡ (lspace I door = isin_l door lock)
            ⟶ ¬(enables I cockpit (Actor a) move)
```

This simple formalization intuitively illustrates how complementary the properties Safety and Security are: the conclusions are negations of each other.

Given the Airplane scenario as defined in the previous section, Safety and Security can be proved. For example, Safety holds for the actor Alice (but similarly also for any other airplane actor). We prove the following lemma.

```
lemma Safety: Safety Airplane_scenario ''Alice''
```

Similarly, in a slightly more complex proof, we can show Security for the pilot Bob (but which also holds for *any* other actor).

```
lemma Security: Airplane_scenario ''Bob''
```

However, even though this simple and short formalization and proof of the Safety and Security properties is feasible, it is not satisfactory. Because of the inbuilt complementarity of the two requirements, in this model, we have both properties and – at the same time – as we could show, the insider attack is still possible. We need to understand the situation better and formalize the attack in more detail to check improvements. We therefore consider the changed scenario which corresponds to a more concrete attack scenario, i.e., the situation when the pilot has moved out of the cockpit. To formalize and analyze this we consider next a re-definition of the infrastructure including its architecture and policies.

E. Refined Attack Scenario

The scenario representing the airplane in danger, has a graph in which the actor Bob, the pilot, is in the cabin rather than the cockpit.

```
ex_graph' ≡  Graph {
 (Node cockpit [Actor ''Charly''], Node door []),
 (Node door [],
  Node cabin [Actor ''Bob'', Actor ''Alice'']) }
```

The credentials of the actors stay the same but the location features' settings now encode that the door is locked.

```
ex_locs' ≡ (λ x. if x = door then (isin_l x lock)
             else (if x = cockpit
                   then (isin_l x air) else True))
```

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

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

Analyzing this new scenario, we first can prove that – as before – the insider attack by Eve is possible, i.e., the global policy does not hold. This means that, fatally, the copilot can move the plane to the ground although the pilot is not in the cockpit.

```
¬ global_policy Airplane_in_danger ''Eve''
```

Can we express the improved version of airplane safety and security relations as described in Section II which requires two authorized personnel to be present at all times in the cockpit? The following adapted set of local policies encodes this. It imposes that in order to perform action `put` at location `cockpit` the number of actors at that node in the graph `G` should be at least 2. The predicate `actors_at` selects the actors list that is part of every node in a graph and the Isabelle inbuilt function `length` produces the number of elements in a list.

```
local_pol_four_eyes G ts ls ≡
(λ y. if y = cockpit then
 {(λ x. (2 ≤ length(actors_at G y)), {put}),
  (λ x. (∃ n. (n @_G door) ∧ Actor n = x ∧
             has (x, ''PIN'')∧
             ls door = isin_l door norm), {move})}
 else (if y = door then {(λ x. True, {move})}
       else (if y = cabin then {(λ x. True,{move})}
             else {})))
```

With these extensions, we can define a scenario with the door on lock and the pilot Bob outside.

```
Airplane_in_danger ≡ Infrastructure ex_graph'
(local_pol_four_eyes ex_graph' ex_creds ex_locs')
 ex_creds ex_locs'
```

The new policy disables any actor in the cockpit to put the position to ground.

```
global_policy Airplane_not_in_danger a
```

F. Extensions to Framework

At this point, we have seen that the Isabelle Insider framework allows to model and analyze the Airplane scenario by using the standard methodology adding a few context specific Safety and Security properties. However, we have also seen that a detailed analysis of the existing and the changed policies necessitates to change to scenario `Airplane_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 `state_transition` and introduce the 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'
```

IV. Discussion and Conclusions

In this paper, we have provided an overview of the development of airplane Safety and Security leading to a model and analysis of an insider attack on an airplane. We have used the Isabelle Insider framework for the formal modeling. The case study has shown that the framework can be applied. It has been possible to express Safety and Security properties in a simple fashion and explore changed policies.

In comparison to previous work by the same authors also in collaboration with others, we have started off investigating Insider threats with the approach of invalidation of security policies in connection with model checking [20], [23]. This early approach also used infrastructure models of organizations, actors and policies but necessarily had to be simpler since model checking does only support finite models. The use of sociological explanation has been pioneered in [24] already with first formal experiments in Isabelle. Finally, the Isabelle Insider framework has been established [1] and has been validated on two of the main three Insider patterns the Entitled Independent and Ambitious Leader. Recently an application to Internet of Things (IoT) Insiders [22] 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 [25] to provide the possibility to refine attacks once they have been identified. This refinement is formalized together with the notion of attack trees. Another extension towards probabilistic modeling using Bayes networks and Markov decision processes has been explored in [26] not yet within the Isabelle framework but using Matlab and the Prism model checker.

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 generalizing 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. In fact, a possible avenue to further explore this extension might lead to an embedding of a reasoning principle to the concepts of model checking. On top of the induction relation, a notion of validity of formulas in a (Kripke)-structure possibly in combination with temporal or other modalities is a possible avenue of future research. Isabelle/HOL supports such extensions, see for example [27]. Additionally we consider game theoretic extensions to the framework following the technical advances made in formalising auctions [28] and modeling [29].

Beyond the purely technical success of the application, a word of caution has been given in the introduction and must be repeated here: necessarily our model abstracts from many details of the application and we can only rely on proved results in as far as we trust that the abstraction only omits irrelevant details. Consistency of proved results with respect to this model is however guaranteed by the Isabelle theorem prover.

In 321 BC, the Samnium general Gaius Pontius [30] set a trap for the Roman army and the Romans had to surrender. As the story goes, he sent for advice to his father what to do. His father answered that he should let the Roman army retreat in honour and make friends with Rome this way. Gaius did not like the idea as it would let off the Romans too lightly and he asked his father again. This time his father answered he should slaughter the whole Roman army of 50,000 soldiers since this would destroy the Roman capability to retaliate for a very long time. Gaius did not like this idea either since it would have meant a big bloodbath. He asked his father a third time for advice and queried about a middle way. His father advised him against this, but against his father's advice Gaius Pontius chose a middle way and dismissed the Romans after publicly humiliating them. They took revenge and Gaius Pontius lost later his life.

Today policy makers still face the dilemma of Gaius Pontius between the two extremes of being fully trusting, or trying to be in complete control. The middle way may still lead to consequences that should be avoided under any circumstance. For instance, if the copilot of the Germanwings flight had not to fear dismissal and in consequence personal economic ruin he could have been more open about his depression and it might not have come to the tragedy. Likewise, if controls had been more stringent and his depression had been communicated to the airline he would probably not have been in control of the aircraft at the time and again it might not have come to the tragedy. This does not offer an answer to regulators or airlines how to handle the dilemma, since a more lenient approach may lead to a higher number of pilots with serious health issues and a more stringent approach may lead to the situation that pilots in need for help would hide their condition. It only summarizes how complicated the matter is and that there are no easy answers. Formalization may, however, help to understand the possible consequences of policy decisions.

References

[1] 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, accepted for publication. [Online]. Available: http://dx.doi.org/10.1109/JSYST.2015.2453215

[2] Wikipedia, "List of aircraft hijackings," accessed December 2015. [Online]. Available: https://en.wikipedia.org/wiki/List_of_aircraft_hijackings

[3] 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)*, 1st ed., ser. SEI Series in Software Engineering. Addison-Wesley Professional, Feb. 2012. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0321812573

[4] 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," in *Proceedings of the third IEEE Workshop on Research in Insider Threats, WRIT'14*. IEEE, 2014.

[5] E. T. Axelrad, P. J. Sticha, O. Brdiczka, and J. Shen, "A bayesian network model for predicting insider threats," in *2013 IEEE Security and Privacy Workshops*. Los Alamitos, CA, USA: IEEE Computer Society, 2013, pp. 82–89.

[6] 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," in *IEEE Security and Privacy Workshops (SPW)*. IEEE, 2014.

[7] D. v. Oheimb, M. Maidl, and R. Robinson, "Security architecture and formal analysis of an airplane software distribution system," in *26th Congress of the International Council of the Aeronautical Sciences (ICAS)*, AIAA, Ed. Proceedings on CD-ROM available from secr.exec@icas.org, 2008, pp. 1–12, http://ddvo.net/papers/ICAS08.html.

[8] 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," in *Proceedings of the third IEEE Workshop on Research in Insider Threats, WRIT'14*. IEEE, 2014.

[9] Wikipedia, "September 11 attacks," accessed January 2016. [Online]. Available: https://en.wikipedia.org/wiki/September_11_attacks

[10] The Star, "Jet cockpit doors nearly impossible to open by intruders," accessed January 2016. [Online]. Available: http://www.thestar.com/news/world/2015/03/26/jet-cockpit-doors-nearly-impossible-to-open-by-intruders.html

[11] "Reinforced cockpit door – description & procedures," September 2002, an Airbus film directed by Bertrand Sirven. Accessed January 2016. [Online]. Available: https://www.youtube.com/watch?v=ixEHV7c3VXs

[12] Wikipedia, "Germanwings flight 9525," accessed January 2016. [Online]. Available: https://en.wikipedia.org/wiki/Germanwings_Flight_9525

[13] F. Kammüller, "Isabelle formalisation of an insider threat framework with examples entitled independent, ambitious leader including attack trees and examples iot and airplane," 2016, available from https://www.dropbox.com/sh/rx8d09pf31cv8bd/AAALKtaP8HMX642fi04Og4NLa?dl=0.

[14] C. G. Hempel and P. Oppenheim, "Studies in the logic of explanation," *Philosophy of Science*, vol. 15, pp. 135–175, April 1948.

[15] H. Esser, *Soziologie – Allgemeine Grundlagen*. Campus, 1993.

[16] F. Kammüller and L. C. Paulson, "A formal proof of sylow's theorem," *Journal of Automated Reasoning*, vol. 23, no. 3, pp. 235–264, 1999.

[17] L. Henrio, F. Kammüller, and M. Rivera, "An asynchronous distributed component model and its semantics," in *Formal Methods for Components and Objects*, ser. LNCS, vol. 5751. Springer, 2009, pp. 159–179.

[18] M. J. Martinko, M. J. Grundlach, and S. C. Douglas, "Toward an integrative theory of counterproductive workplace behaviour," *International Journal of Selection and Assessment*, vol. 10, no. 1–2, pp. 36–50, 2002.

[19] B. Marcu and H. Schuler, "Antecedents of counterproductive behaviour at work: a general perspective," *Journal of Applied Psychology*, vol. 89, no. 4, p. 647, 2004.

[20] F. Kammüller and C. W. Probst, "Combining generated data models with formal invalidation for insider threat analysis," in *IEEE Security and Privacy Workshops (SPW)*. IEEE, 2014.

[21] F. Kammüller, M. Wenzel, and L. C. Paulson, "Locales - a sectioning concept for isabelle," in *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99*, ser. LNCS, Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, , and L. Thery, Eds., vol. 1690. Springer, 1999.

[22] 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 - Fourth International Conference, HAS 2015, Held as Part of HCI International 2016, Toronto*, ser. Lecture Notes in Computer Science. Springer, 2016, invited paper.

[23] F. Kammüller and C. W. Probst, "Invalidating policies using structural information," in *WRIT'13*. IEEE, 2013.

[24] 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.

[25] C. W. Probst, F. Kammüller, and R. R. Hansen, "Formal modelling and analysis of socio-technical systems," in *Nielsens Festschrift*, ser. LNCS. Springer, 2016, in print.

[26] T. Chen, F. Kammüller, I. Nemli, and C. W. Probst, "A probabilistic analysis framework for malicious insider threats," in *Human Aspects of Information Security, Privacy, and Trust - Third International Conference, HAS 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015. Proceedings*, ser. Lecture Notes in Computer Science, T. Tryfonas and I. G. Askoxylakis, Eds., vol. 9190. Springer, 2015, pp. 178–189. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-20376-8_16

[27] S. Helke and F. Kammüller, "Verification of statecharts using data abstraction," *International Journal of Advanced Computer Science and Applications*, 2016, to appear.

[28] D. Liu, X. Wang, and J. Camp, "Game-theoretic modeling and analysis of insider threats," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 75–80, 2008.

[29] M. B. Caminati, M. Kerber, C. Lange, and C. Rowat, "Sound auction specification and implementation." ACM, 2015.

[30] "Gaius Pontius, Sabine leader," 2015, accessed Janary 2016. [Online]. Available: http://www.unrv.com/bio/gaius-pontius.php

[31] *Proceedings of the third IEEE Workshop on Research in Insider Threats, WRIT'14*. IEEE, 2014.