Attack Tree Analysis for Insider Threats on the IoT using Isabelle

Florian Kammüller[1], Jason R.C. Nurse[2], and Christian W. Probst[3]

[1] Middlesex University London, f.kammueller@mdx.ac.uk
[2] University of Oxford, jason.nurse@cs.ox.ac.uk
[3] Technical University Denmark, cwpr@dtu.dk

Abstract. The Internet-of-Things (IoT) aims at integrating small devices around humans. The threat from human insiders in "regular" organisations is real; in a fully-connected world of the IoT, organisations face a substantially more severe security challenge due to unexpected access possibilities and information flow. In this paper, we seek to illustrate and classify insider threats in relation to the IoT (by 'smart insiders'), exhibiting attack vectors for their characterisation. To model the attacks we apply a method of formal modelling of Insider Threats in the interactive theorem prover Isabelle. On the classified IoT attack examples, we show how this logical approach can be used to make the models more precise and to analyse the previously identified Insider IoT attacks using Isabelle attack trees.

1 Introduction

Insider threats are notoriously difficult to prevent since the usual method of introducing a security perimeter and identifying actor's capabilities are useless: any actor possibly having privileges and access can turn into an attacker and hit organisations where it hurts most. 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.

The combination of IoT and insider threat thus represents a highly risky area in terms of security, privacy, and trust. This problem has been highlighted and discussed in detail in some of our recent work on the 'smart insider' [16]. The aim of this paper is to extend that work and propose a set of rigorous modelling techniques in order to provide a better understanding of IoT-facilitated insider-threat scenarios and related architectures; this would also cover related analysis methods to verify security properties of a given model and its policies. The IoT aims at integrating small devices around humans. A corresponding paradigm corresponding to the IoT might be entitled "human centric computing". Research work on insider threat on the other side has revealed that the consideration of the human aspect is crucial to arrive at useful security models. Frameworks for insider threats [15] take human aspects in consideration to provide models for a characterisation of the tipping point when an insider turns into a threat. Logical modeling and analysis techniques use such taxonomies and a three-step process of social explanation [13] to provide machine supported verification techniques of infrastructure models and their policies.

In this paper, we start by reflecting on existing research into insider threat and then consider work on attack vector models for the IoT case [16] (Section 2). We use a framework for modelling of Insider threats in logic with the interactive proof assistant Isabelle [13] to formalize the IoT scenarios and express these attack vectors enabling the machine supported proof of security properties (Section 3). The formalisation of attack vectors provides a formal underpinning that enables a stepwise refinement into given IoT insider scenarios. We demonstrate this on a real IoT insider attack case. Finally, we conclude the paper and present avenues for future work (Section 4).

2 Reflecting on the IoT Insider Threat Challenge

The threat that insiders pose to organisations is well-known and has been documented in industry, academic and the media. In a recent survey, globally 89% of respondents felt that their organisation was now more at risk than before to an insider attack [22]. The key challenge with insiders is that they have an elevated level of access in order to do their jobs, but such access could easily be abused to exploit or harm their employer. In literature, three main types of insider threat are commonly acknowledged. These include insider fraud (*i.e.*, abusing one's privileges in the company for personal gain), sabotage (*i.e.*, destruction of property) and Intellectual Property (IP) theft (*i.e.*, stealing commercially sensitive enterprise data) [3]. Insiders can also be considered from the context of their attack, that is, whether it is intentional or accidental. There has been significant research on the former topic in the past, but work on accidental insiders is slowly gaining traction.

As the emphasis on the insider threat has grown, so too have the approaches to prevent and detect it. There have been proposals outlining models and frameworks for understanding and reasoning about insider threat [8,15]. Other research has focused on prevention and prediction to allow more proactive responses to the problem [6,7]. Moreover, many systems have been put forward specifically to detect behaviour of threatening insiders [1,4]. Commonly, such systems aim at areas such as monitoring activity on corporate networks (*e.g.*, emails, file accesses), analysing online activity data, and incorporating psychological information.

Possibly one of the most significant challenges with insider threats is the variety of ways in which they can attack their employers. For instance, sensitive IP can be exfiltrated via printing, email, FTP, remote access, or pen drives. Systems can be sabotaged using several physical or electronic means (*e.g.*, deletion, malware) by insiders. Also, insiders could engage in various fraudulent activities at the expense of their employers. To add to this challenge, we are quickly progressing towards a society where every day objects (*e.g.*, phones, watches, cars) are connected in the Internet-of-Things.

While the IoT has been discussed for over a decade, only now are we starting to witness substantial progress towards technology standardisation, along with a variety of different connected products reaching businesses and homes. As these technologies enter businesses and offices more, however, each enterprises' attack surface is significantly enlarged. For instance, cameras on smart-watches worn by insiders can take discrete pictures of sensitive IP, thereby allowing exfiltration of data with no digital trace on corporate systems [16]. This lack of digital trace makes it extremely difficult for current systems to detect smart insiders, given monitoring does not currently encompass such personal devices. Having discussed the challenges faced by such insiders, next we briefly reflect on a few key attack vectors to explore the problem in greater detail.

2.1 Attack Vectors of Insiders using IoT Devices

As mentioned above, insiders using the IoT represent a significant challenge for enterprises. In our previous work, we have assessed this problem in detail, and outlined several vectors through which insiders may attack their employers [16]. To structure that work, we drew on the VERIS 4A approach to define cyber attacks [21]. 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 8+8 attack vectors (AVs) from [16] 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. Picking one AV from each group, we use them as a representative basis for analysis and discussion throughout the remainder of this paper.

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 [20]. 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.

3 Formal Model of Insider IoT Threats in Isabelle

Motivation of attackers and the behavioural aspects exhibited during attacks is often not considered, *e.g.*, in detection systems for cyber security (as discussed above). The challenge we approach here is to accommodate insiders' behaviour into formal models for IoT insider attacks using logical modelling and analysis techniques.

3.1 Social Explanation for Insider Threats in Isabelle

Previous work [13] uses the process of sociological explanation based on Max Weber's *Grundmodell* and its logical interpretation to explain insider threats by moving between societal level (macro) and individual actor level (micro). The interpretation into a logic of explanation is formalized in Isabelle's Higher Order Logic thereby providing a tool to prove global security properties with machine assistance [13]. Isabelle/HOL is an interactive proof assistant based on Higher Order Logic (HOL). It enables specification of so-called object-logics for an application. Examples reach from mathematical theory [10] to component based software engineering [5]. Object-logics comprise new types, constants and definitions and reside in theory files, *e.g.*, the file `Insider.thy` contains the object-logic we define for social explanation of insider threats below. We construct our theory as a conservative extension of HOL guaranteeing consistency, *i.e.*, we do not introduce new axioms that could lead to inconsistencies.

In this paper, we show how the 4A modelling approach (see Section 2.1) can be accommodated by this insider threat theory and a suitable extension to attack vectors. We first provide here only the elements of this Insider theory necessary as a basis for attack trees and for modelling IoT applications. For a more complete view, please refer to [13] and the related online Isabelle resources [9].

In the Isabelle/HOL theory for Insiders, we express policies over actions `get`, `move`, `eval`, and `put` representing the *Actions* category from the 4As (see Section 2.1). We abstract here from concrete data – actions have no parameters:

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

The next of the 4As is the *Actor* which is represented by an abstract type and a function that creates elements of that type from identities:

```
typedecl actor
type_synonym identity = string
consts Actor :: string ⇒ actor
```

Policies describe prerequisites for actions to be granted to actors given by pairs of predicates (conditions) and sets of (enabled) actions:

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

We integrate policies with a graph into the infrastructure providing an organisational model where policies reside at locations and actors are adorned with additional predicates to specify their 'credentials':

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

These local policies serve to provide a specification of the ‘normal’ behaviour of actors but are also the starting point for possible attacks on the organisation’s *Assets*. The assets are defined by the goals of the attacks, *i.e.*, the roots of the attack trees (see below). The `enables` predicate specifies that an actor `a` can perform an action `a'∈ e` at location `l` in the infrastructure `I` if `a`’s credentials (stored in the tuple space `tspace I a`) imply the location policy’s (stored in `delta I l`) condition `p` for `a`:

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

3.2 Attack Trees in Isabelle

We now extend the theory Insider by Attack trees by defining the base attacks and how they constitute an attack sequence. This corresponds to combining Actions into an “insider-attack vector” (see Section 2.1). We represent these in Isabelle/HOL as a data type and a list over this datatype:

```
datatype baseattack = None | Goto "location"
                    | Perform "action" | Credential "location"
type_synonym attackseq = "baseattack list"
```

The following definition `attree`, defines the nodes of an attack tree. The simplest case is when a node in an attack tree is an *Asset*, *i.e.*, a base attack as defined above. Attacks can also be combined as the “and” of other attacks. The third element of type `attree` is a `baseattack` (usually a `Perform action`) that represents this attack, while the first element is an attack sequence and the second element is constituted by the fourth of the 4As, the *Attribute*. As described in Section 2.1, an attribute describes the impact of the attack on the asset in a variety of ways. We therefore allow a great degree of freedom in our logical model by modeling this attribute as an element of type “string”:

```
datatype attree = BaseAttack "baseattack" ("𝒩 (_)") |
                  AndAttack "attackseq" "string" "baseattack" ("_ ⊕∧^(_) _")
```

As the corresponding projection functions for `attree` we define `get_attseq` and `get_attack` returning the entire attack sequence or the final base attack, respectively.

The following inductive predicate `UI_AV7_intro` represents the attack vector **UI-AV7** (see Section 2.1) by an inductive definition. It formalizes how the attacker intercepts the traffic between the smart devices by moving into close range and getting thus the data. Logically, this is justified if an actor `a` can get data at location `l` in the extended infrastructure `add_credential I a s` in which he possesses the credential `s` – as is expressed by the second `enables` proviso:

```
⟦ enables I l a move; enables (add_credential I a s) l a get ⟧
⟹ UI_AV7 I s
   (get_attackseq ([Goto l, Perform get] ⊕∧^(move−intercept) Credential l))
   (Credential l)
```

An attack tree is constituted from the above defined nodes of type `attree` but children nodes must be refinements of their parents. Refinement means that some portion of the attack sequence has been extended according to rules like the above `get_then_move`. Similar rules can be defined for all of the attack vectors for malicious insiders MI_AVi and for unintentional insiders UI_AVi [16]. Higher Order Logic allows us to define a set over the inductive predicates that we defined as attack vectors above. We can then assemble all attack vectors in the following set (we omit all but the two we illustrate here for brevity):

```
definition attack_vectors::
([infrastructure, string, attackseq, baseattack] ⇒ bool)set
where attack_vectors ≡ {MI_AV4, UI_AV7}
```

We formalize the constructor relation for the refinement of attack trees by the following inductive predicate `refines_to` syntactically represented as the infix operator $\sqsubseteq$. The rules `trans` and `refl` make the refinement a preorder; the rule `refineI` shows how attack vectors from the previously defined set can be integrated into the refinement process. If we replace the attack `a` by a sequence `l` of an attack vector from our predefined set of attack vectors `P`, we refine the attack sequence `A` into `A'` (the auxiliary function `sublist_rep` replaces symbol `a` in list `l` by a list, here `get_attseq A`):

```
inductive
refines_to :: "[attree, infrastructure, attree] ⇒ bool" ("_ ⊑_(_) _")
where
refineI: ⟦ P ∈ attack_vectors; P I s l a;
              sublist_rep l a (get_attseq A) = (get_attseq A');
              get_attack A = get_attack A' ⟧ ⟹ A ⊑_I A' |
trans: ⟦ A ⊑_I A'; A' ⊑_I A'' ⟧ ⟹ A ⊑_I A'' |
refl : A ⊑_I A
```

The refinement of attack sequences allows the expansion of top level abstract attacks into longer sequences. Ultimately, we need to have a notion of when a sufficiently refined sequence of attacks is valid. This notion is provided by the final inductive predicate `is_and_attack_tree`. It integrates the base cases where base attacks can be directly logically derived from corresponding enables properties; it states that an attack sequence is valid if all its constituent attacks are so and it allows to transfer validity to shorter attacks if a refinement exists:

```
inductive
is_and_attack_tree :: [infrastructure, actor, attree] ⇒ bool ("_, _ ⊢ _")
where
att_act: enables I l a a' ⟹ I , a ⊢ 𝒩(Perform(a')) |
att_goto: enables I l a (move) ⟹ I, a ⊢ 𝒩(Goto l) |
att_cred: enables I l a (get) ⟹ I, a ⊢  𝒩(Credential l) |
att_list: ⟦ ∀ a ∈ (set(as)). I, a' ⊢ 𝒩(a) ⟧ ⟹ I, a' ⊢ as ⊕_∧^s a'' |
att_ref: ⟦ A ⊑_I A'; I, a ⊢ A' ⟧ ⟹ I, a ⊢ A
```

The Isabelle/HOL theory library provides many list functions. We use these to define the "or" of attack trees by folding the above validity over a list of attacks:

`I, a` $\vdash_{G \oplus_\vee}$ `al` $\equiv$ `fold (`λ `x y. (I, a` $\vdash$ `x)` $\vee$ `y) al False`

To validate this formalisation of the attack trees, we now show how a scenario attack can be derived, based on an extended example from our previous work [16].

3.3 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 [2], 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.

3.4 Application of Insider Theory and Attack Trees to Example

For the application to the office scenario, we only model two identities, `Boss` and `Employee` representing an employee and his boss. We define the set of office actors as a local definition in the locale `scenarioOffice`. We show here in a first instance the full Isabelle/HOL syntax but in all subsequent definitions we omit the `fixes` and `defines` keywords and also drop the types for clarity 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''}
```

The graph representing the infrastructure of the office case study contains only the minimal structure: (1) employee's office, (2) boss's office, (3) smartphone:

```
  office_locations ≡ {Location 1, Location 2, Location 3}
```

The global policy is 'no one except office actors can get anything from the boss's office':

```
  global_policy I a ≡  a ∉ office_actors ⟶
                      ¬(enables I (Location 2) (Actor a) get)
```

Next, we have to provide the definition of the infrastructure. We first define the graph representing the organisation's locations and the positions of its actors. Locations are wrapped up with the datatype constructor `NL` and actors using the corresponding constructor `NA` to enable joining them in the datatype `node` and thus creating the following `node graph` as a set of pairs between locations or actors:

```
  ex_graph ≡ Graph {(NA (''Boss''), NL (Location 2)),
                    (NL (Location 2), NL(Location 1)),
                    (NL (Location 2), NL(Location 3)),
                    (NA (''Employee''), NL (Location 1))}
```

Policies are attached to locations in the organisation'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 credentials which are defined as predicates over actors, *i.e.*, boolean valued functions describing, for example, whether an actor inhabits a role, or, whether an actor possesses something, like an identity or a key. The second elements are sets of actions that are authorised in this location for actors authenticated by the credentials:

```
local_policies ≡
(λ x.  if x = Location 1 then
 {(λ x. (ID x ''Boss'')∨(ID x ''Employee''),{get,put}),(λ x.True,{move})}
   else (if x = Location 2 then
    {((λ x. has (x, ''PIN'')), {get,put}), (λ x. True, {move})}
       else (if x = Location 3 then
        {((λ x. True, {get,put,move}))}
          else {})))
```

The final component of any infrastructure is the credentials contained in a `tspace`. We define the assignment of the credentials to the actors similarly as a predicate over actors that is true for actors that have the credentials:

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

Finally, we can put the graph, the local policies, and the credential assignment into an infrastructure:

```
Office_scenario ≡ Infrastructure ex_graph local_policies ex_creds
```

Note, that all the above definitions have been implemented as local definitions using the locale keywords `fixes` and `defines` [14]. 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_inv2`:

```
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 overall procedure corresponds to the approach of invalidation of a global policy based on local policies for a given application scenario [11].

However to systematically derive the actual attack for the Employee blackmail we next apply the attack vector analysis presented in Section 3.2. First, we prove the following `move_intercept_lem` property:

```
lemma move_intercept_lem: UI_IV7 Office_scenario ''PIN''
  (get_attseq ([Goto (Location 2), Perform get] ⊕∧^{move intercept} Perform get))
  (Credential (Location 2))
```

After reducing with the defining rule `UI_AV7_intro` (see Section 3.2), the proof requires solving two "enables" subgoals; the final one uses the `add_credential` for `Employee`. This lemma immediately implies the following refines property:

([Credential (Location 2)] $\oplus_{\wedge}^{move-intercept}$ Perform get)
$\sqsubseteq_{Office-scenario}$
([Goto (Location 2), Perform get)] $\oplus_{\wedge}^{move-intercept}$ Perform get)

At this point we have constructed an attack tree as depicted in Figure 1. As

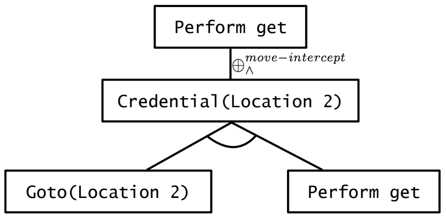

Fig. 1. Attack tree refines high level attack into base attacks.

a final step of verification, we show that the refined attack at the leaves of the tree is valid, *i.e.*, each step in it is a possible base attack in the scenario (see Section 3.2):

```
lemma final_attack: Office_scenario, Actor ''Employee'' ⊢
([Goto (Location 2), Perform get] ⊕∧^{move-intercept} Perform get)
```

The last lemma together with the refinement gives us finally that the top level abstract attack is a valid attack:

```
theorem office_attack: Office_scenario, Actor ''Employee'' ⊢
([Credential (Location 2)] ⊕∧^{move-intercept} Perform get)
```

4 Discussion and Conclusion

In this paper we have presented an approach to characterising malicious and unintentional insider threats on the IoT by attack vectors. We added precision to a tentative taxonomy [16] by using a logic based Insider threat model [13] in Isabelle. We illustrated its use on the IoT Insider case of an employee blackmailing his boss with communication data intercepted from a smart-watch/smartphone with weak security. An extension of the Isabelle framework for insider threats to attack trees enabled the logic representation of the malicious and unintentional attack vectors. We summarized all attack vectors in a set. A notion of refinement of attack trees allows to apply the attack vectors in this set as possible refinement steps to a high level attack. The refinement serves for analysing attacks: if a high level attack can be sufficiently refined, a notion of validity of attack permits to finalise the attack analysis by proving an Isabelle theorem. The extended Isabelle framework has been introduced and illustrated on the Employee blackmail case study.

It must be clearly stated, that the attack tree generation is not a fully automated process. Isabelle is an interactive proof assistant. That is, the attack and the refinement have to be input by the user and the refinement and validity theorems have to be proved in an interactive process. However, the malicious and unintentional attack vectors provide a set of possible high level attacks that can be used as starting points for an attack tree refinement in the Isabelle Insider framework. This process supports systematic tool based analysis of infrastructures for Insider threats revealing weaknesses in policies and exemplifying the attack vectors. Furthermore, the demonstrated application to an IoT insider case shows that the proof obligations of refinement and validity can be achieved by a short series of applications of automated proof tactics that are integrated into Isabelle (for illustration see the online resources [9]).

A pioneering effort to assess insider attacks was the CMU-CERT Insider Threat project [3]. *Attack trees* as specified in [18] define the attacker's main goal as the root of a tree; this goal is then disjunctively or conjunctively refined into sub-goals until the reached subgoals represent basic actions that correspond to atomic components. Disjunctive refinements show up alternative pathways to achieving a goal, whereas conjunctive refinements visualize the attack steps leading to a goal. Automated generation of attack graphs mostly considers computer networks only [17,19]. These techniques usually start by specifying atomic attacks. By contrast, our approach is based on [12]: the attack consists in invalidating a policy, and the model just provides the infrastructure and methods for deriving the attack tree.

The presented work illustrates that the logic based approach including the human factor into insider threat modelling and analysis [13] extends also to the security critical domain of IoT Insiders. Further experimentation with the provided framework in planned projects will focus on integrating with quantitative analysis.

References

1. A. Ambre and N. Shekokar. Insider threat detection using log analysis and event correlation. *Procedia Computer Science*, 45:436–445, 2015.
2. 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`.

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)*. Addison-Wesley Professional, 2012.
4. G. Gavai, K. Sricharan, D. Gunning, R. Rolleston, J. Hanley, and M. Singhal. Detecting insider threat from enterprise social and online activity data. *ACM CCS International Workshop on Managing Insider Security Threats*. ACM, 2015.
5. L. Henrio, F. Kammüller, and M. Rivera. An asynchronous distributed component model and its semantics. *FMCO. LNCS* **5751** Springer, 2009.
6. S. Hoyer, H. Zakhariya, T. Sandner, and M. H. Breitner. Fraud prediction and the human factor: An approach to include human behavior in an automated fraud audit. *45th Hawaii International Conference on System Science (HICSS)*, IEEE, 2012.
7. U. Hugl. Putting a hat on a hen? Learnings for malicious insider threat prevention from the background of german white-collar crime research. In *Human Aspects of Information Security, Privacy, and Trust*, LNCS **9190** Springer, 2015.
8. J. Hunker and C. W. Probst. Insiders and insider threatsan overview of definitions and mitigation techniques. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, 2(1):4–27, 2011.
9. F. Kammüller. Isabelle Insider framework with examples, 2015. Available at `https://www.dropbox.com/sh/rx8d09pf31cv8bd/AAALKtaP8HMX642fi04Og4NLa?dl=0`.
10. F. Kammüller and L. C. Paulson. A formal proof of sylow's theorem. *Journal of Automated Reasoning*, 23(3):235–264, 1999.
11. F. Kammüller and C. W. Probst. Invalidating policies using structural information. In *WRIT'13*. IEEE, 2013.
12. F. Kammüller and C. W. Probst. Combining generated data models with formal invalidation for insider threat analysis. *IEEE Security and Privacy Workshops (SPW), WRIT'14*, IEEE 2014.
13. F. Kammüller and C. W. Probst. Modeling and verification of insider threats using logical analysis. *IEEE Systems Journal*. In print, 2016.
14. 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.
15. 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), WRIT'14*, IEEE, 2014.
16. 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.
17. C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. *Workshop on New security paradigms, NSPW'98*, 1998.
18. C. Salter, O. S. Saydjari, B. Schneier, and J. Wallner. Toward a secure system engineering methodology. *Workshop on New Security Paradigms, NSPW'98*, 1998.
19. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. *IEEE Symposium on Security and Privacy (S&P'02)*, IEEE, 2002.
20. Symantec. How safe is your quantified self?, 2014. Tech. Rep.
21. VERIS. Veris: The vocabulary for event recording and incident sharing, 2015. Available from `http://veriscommunity.net`.
22. Vormetric. 2015 vormetric insider threat report, 2015. Available from `http://www.vormetric.com/campaigns/insiderthreat/2015/`.