predict
prioritise
prevent

TRESPASS

Technology-supported Risk Estimation by Predictive Assessment of Socio-technical Security

Deliverable D1.3.4

TRESPASS socio-technical security model and specification languages

Project:	TRESPASS
Project Number:	ICT-318003
Deliverable:	D1.3.4
Title:	TRESPASS socio-technical security model and specification languages
Version:	1.0
Confidentiality:	Public
Editor:	Christian W. Probst
Cont. Authors:	J.W. Bulée, M. Ford, M. Fraile, O. Gadyatskaya, R.R. Hansen, D. Ionita, H. Jonkers, L. Montoya, C.W. Probst, A. Tanner, A.S. Yesuf
Date:	2016-10-31

Part of the Seventh Framework Programme
Funded by the EC-DG CONNECT

Members of the TRESPASS Consortium

1.	University of Twente	UT	The Netherlands
2.	Technical University of Denmark	DTU	Denmark
3.	Cybernetica	CYB	Estonia
4.	GMV Portugal	GMVP	Portugal
5.	GMV Spain	GMVS	Spain
6.	Royal Holloway University of London	RHUL	United Kingdom
7.	itrust consulting	ITR	Luxembourg
8.	Goethe University Frankfurt	GUF	Germany
9.	IBM Research	IBM	Switzerland
10.	Delft University of Technology	TUD	The Netherlands
11.	Hamburg University of Technology	TUHH	Germany
12.	University of Luxembourg	UL	Luxembourg
13.	Aalborg University	AAU	Denmark
14.	Consult Hyperion	CHYP	United Kingdom
15.	BizzDesign	BD	The Netherlands
16.	Deloitte	DELO	The Netherlands
17.	Lust	LUST	The Netherlands

Document History

Authors		
Partner	Name	Chapters
AAU	René Rydhof Hansen	2, 3
BD	Henk Jonkers	3
DTU	Christian W. Probst	ALL
CHYP	Margaret Ford	4
GMVS	Marlon Fraile	4
GUF	Ahmed S. Yesuf	4
IBM	Axel Tanner	4
UL	Olga Gadyatskaya	3
UT	Dan Ionita Jan-Willem Bulée Lorena Montoya	3 2 2

Quality assurance		
Role	Name	Date
Editor	Christian W. Probst	2016-10-31
Reviewer	Jan Willemson	2016-10-15
Reviewer	Dieter Gollmann	2016-10-15
WP & Task leader	Christian W. Probst	2016-10-31
Coordinator	Pieter Hartel	2016-10-31

Circulation	
Recipient	Date of submission
Project Partners	2016-10-25
European Commission	2016-10-31

Acknowledgement: *The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TRESPASS). This publication reflects only the authors' views and the Union is not liable for any use that may be made of the information contained herein.*

Contents

List of Figures

List of Tables

Management Summary

Key takeaways:

- We have identified relevant, security-related properties of socio-technical systems and best practices of existing system models for assessing and managing IT security-related risks.
- The TRE$_S$PASS socio-technical security model combines these best practices, and provides the necessary abstractions to model these properties and makes them available for tools and analyses.
- The final version of the TRE$_S$PASS socio-technical security model has been validated on case studies from different domains.

The primary goal of the TRE$_S$PASS project is to develop tools that facilitate assessment and management of IT security-related risks in an organisation, spanning both technological and sociological issues. To support this goal, we have developed a robust, yet flexible, modelling formalism that can capture and unify such diverse aspects of organisations as IT infrastructure, organisational structure, and physical structures. This modelling formalism forms the basis for identifying possible attacks on the organisation.

The TRE$_S$PASS socio-technical security model and its specification language have been developed based on a study of relevant, security-related properties of socio-technical systems and best practices of existing system models for assessing and managing IT security-related risks. The model is based on a new calculus that represents actors and processes as nodes that move around and can contain other assets such as data and items; this approach unifies the way how these assets and processes are dealt with. The model forms the central abstraction of socio-technical systems in the TRE$_S$PASS process and tools, and provides access to the represented socio-technical system.

In this deliverable, we describe the final version of the TRE$_S$PASS socio-technical security model and specification language, show the overall structure and decisions made in developing the model, and present the modelling of some case studies investigated in the project.

1. Introduction

System models have found many applications in analysing organisations for their vulnerability to different kinds of threats. To support this analysis of organisations, several system models have been introduced that model organisations' infrastructure and actors. Examples of such models include ExASyM (Probst & Hansen, 2008), Portunes (Dimkov, Pieters, & Hartel, 2010), and ANKH (Pieters, 2011a). These and other models have been surveyed for developing the TRE$_S$PASS socio-technical security model and specifying its requirements (The TRE$_S$PASS Project, D1.1.2, 2015). All these models follow similar ideas, namely the modelling of infrastructure and data, and analysing the modelled organisation for possible attacks. The main differences are in representation and the level of detail with which parts of the model system can be represented.

This deliverable documents the final TRE$_S$PASS socio-technical security model and its representation. We also briefly discuss some alternative modelling approaches investigated in the project, and the insights they have provided. To illustrate some best practices in modelling, we show the model's application to case studies from the project.

1.1. Goals

The goals of this deliverable are to:

- Present the final TRE$_S$PASS socio-technical security model;
- Discuss related modelling approaches that have been pursued in the TRE$_S$PASS project; and
- Showcase the application of the model to the case studies.

1.2. Choices made

The TRE$_S$PASS socio-technical security model has been designed based on a structured literature review documented partly here and in (The TRE$_S$PASS Project, D1.1.2, 2015).

For space reasons only models for three case studies are shown to illustrate some of the components of models and some best practices.

1.3. Foreground and Background

The TRESPASS socio-technical security model is foreground IP, so is the STS, the e3 model, and the models developed for the case studies.

1.4. Document Structure

Chapter 2 presents the conclusions of the structured literature review of socio-technical security models and presents the TRESPASS model. Chapter 3 presents alternative modelling approaches that have been pursued in the project, and illustrates the motivation for doing so. After presenting models for some case studies in Chapter 4, the deliverable closes with concluding remarks in Chapter 5.

2. The TRESPASS Model

This chapter presents the final version of the TRESPASS socio-technical security model for representing socio-technical systems.

To identify properties and shortcomings of existing models, we have performed a structured literature review for socio-technical systems. The next section summarises the findings of this review and discusses how the TRESPASS socio-technical security model combines best practices. This is followed by a description of the components of the TRESPASS model in Section 2.2 and the description of the underlying process calculi and its semantics in Section 2.3.

2.1. Requirements for Socio-Technical Security Models

This section presents the high-level findings of a structured literature review in the area of security modelling, to identify the requirements for the TRESPASS socio-technical security model. The final requirements and the complete results of the review can be found in (The TRESPASS Project, D1.1.2, 2015). Whenever we refer to studies this relates to the papers presented in (The TRESPASS Project, D1.1.2, 2015). Table 2.1 presents the summarised findings of (The TRESPASS Project, D1.1.2, 2015).

2.1.1. Representation

The studies investigated use either graphical or textual model representations. The graphical representations can be divided in trees, graphs, diagrams, and maps, with the former two having clear mathematical properties that support formal analyses (Probst, Kammüller, & Hansen, 2016). Diagrams and maps also help to communicate models to tool users. Plain textual representations are mainly used as input for tools, and support also some form of graphical representation. An optimal model representation is therefore a hybrid of these approaches.

The TRESPASS model supports an XML-based representation; this textual representation can be visualised by standard tools and especially the WP4 visualisations and the Attack Navigator Map.

2.1.2. Infrastructure

All of the studies investigated directly addressing modelling approaches are able to model the digital or virtual layer of the infrastructure. About half of them model the physical infrastructure as well. Only few studies reflect the social layer of the system infrastructure. In addition, most of the studies model only the attacker who traverses the system acquiring knowledge and/or access on the way.

The TRE$_S$PASS model needs to model all three layers, so it will be able to adapt to the existing approaches.

2.1.3. Assets and Containment

All models support assets on the different levels considered. Some studies only consider information, all other studies include physical and to some extent digital artefacts.

Containment is only addressed by a few studies. The studies that are able to model an actor at a location, are also able to model the nesting of objects. Some studies provide a different notion of containment in the sense of annotation/quantification properties, e.g., hosts containing vulnerabilities.

The TRE$_S$PASS model can represent assets such as data and items. The model represents containment through location attributes; containment can be arbitrarily deep.

2.1.4. Processes, Actions, and Behaviour

All studies agree on the definition of a process as a sequence of steps. Most studies only mention the existence of processes as part of the model. However, some models provide explicit support for processes.

In most studies, actions relate to the digital domain varying from specific actions as Read and Out to “all computing” in general.

Behaviour relates to the social domain, where human actors perform actions within the model. Human behaviour involves all attacker behaviour that is needed to achieve a goal and more specific behaviour like moving between locations start or start a process on a computer. Using human behaviour as a general, non restricted concept provides flexibility. On the other hand, a restricted set of actions enables formal treatment and analysis. A trade-off should allow the freedom to model human behaviour in a proper way, and also be able to be formally checked.

Also the TRE$_S$PASS socio-technical security model has natural support for processes through the underlying process calculus, and supports actions and behaviour through this underlying process calculus, that also encodes actor behaviour.

2.1.5. Actors

Most of the studies focus on insiders as they have better access and knowledge. Some of the studies do not distinguish between insiders and outsiders. Few studies do not even distinguish between humans and non-humans in the modelling phase. In real life attackers can collaborate; only few of the studies are able to represent this. One study does not even model actors or a potential attacker, but analyses the context for possible information leakage without a specific attack scenario involving an actor.

In the TRE$_S$PASS model, insiders and outsiders are the same. They differ only in their knowledge about the organisation.

2.1.6. Policies

Most of the studies take into consideration only low level policies in terms of accessibility/reachability. While considering high level (organisational policies) is essential, it could also be problematic in case of inconsistencies between low and high level policies. One study pays close attention to this problem and derives attack scenarios from analysing the policies on different levels.

The TRE$_S$PASS model must enable the reasoning about policies and their relationship; especially contradictions between and holes in policies are of interest, since they may enable attacks.

2.1.7. Quantitative Measures

A number of studies support quantitative annotations to annotate the model and attack. Those that are most frequently supported are: probability that an attack will succeed and the costs of an attack. The supported annotations are mainly properties of an attacker or actions, e.g., required skill and risk of detection, however there are also some organisational properties, e.g., impact of an attack.

The TRE$_S$PASS model supports quantitative annotations through unique identifiers for all model elements. These identifiers enable the mapping from elements to properties through the knowledge base.

2.1.8. Attacks, Vulnerabilities, and Countermeasures

Only few of the studies provide vulnerabilities as input to the model. Usually they are described by domain experts and are based on previous attacks. The most frequently used attack representation is a sequence of actions. While it gives a high level overview and is easy to understand by non-experts, it has a flat structure and thus does not provide many details. The other popular representation are attack trees, presenting threats in a hierarchical structure; however, they loose the notion of sequential actions. Another

disadvantage of attack trees (or in some studies referred to as attack patterns) is that they cannot reflect interactions between different attacks. Only two studies model attacks carried out by more than one attacker, i.e., by collaborating attackers.

The TRE$_S$PASS model represents vulnerabilities and counter measures through qualitative properties in the knowledge base.

Study	Representation	*Physical* Infrastructure	*Digital* Infrastructure	*Social* Infrastructure	Assets	Containment	Processes	Actions	Behaviour	Actors	*Low Level* Policies	*High Level* Policies	Quantitative Measures	Attacks	Vulnerabilities	Countermeasures
TRE$_S$PASS socio-technical security model	Multi	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
(Ten, Manimaran, & Liu, 2010)	ADtree	-	+	-	+	-	+	+	-	-	-	-	+	+	+	+
(Xie, Chen, Wang, Chen, & Hu, 2009)	Graph	-	+	-	+	-	+	-	-	-	+	-	+	+	+	-
(Shahriari, Makarem, Sirjani, Jalili, & Movaghar, 2010)	Text	-	+	-	+	-	-	+	-	-	-	-	-	+	-	-
(Karpati, Opdahl, & Sindre, 2011)	Multi	-	+	-	+	-	+	+	+	+	-	-	+	+	+	-
(Dragovic & Crowcroft, 2004, 2005; Dragovic, 2006)	Tree	+	+	-	+	+	-	-	-	-	+	-	+	+	-	+
(Probst, Hansen, & Nielson, 2007; Probst & Hansen, 2008)	Graphical	+	+	+	+	-	+	+	+	+	+	-	-	+	-	+
(Franqueira, Lopes, & van Eck, 2009)	Multi	+	+	+	+	+	-	+	+	+	+	-	+	+	-	-
(Mathew et al., 2005; Mathew, Upadhyaya, Ha, & Ngo, 2008)	DAG	+	+	-	+	+	-	-	-	+	+	-	-	±	-	±
(Samarji, Cuppens, Cuppens-Boulahia, Kanoun, & Dubus, 2013)	Graph	-	+	-	+	-	±	-	-	+	±	-	-	+	-	-
(Sommestad, Ekstedt, & Johnson, 2010; Sommestad, Ekstedt, & Holm, 2012)	UML like	-	+	-	+	-	+	+	-	-	-	-	+	+	+	+
(Scott, 2004)	Tree	+	+	+	-	+	+	+	-	+	+	-	-	-	-	-
(Dimkov et al., 2010; Dimkov, 2012)	DAG	+	+	+	+	+	+	+	+	+	+	+	-	+	-	-
(Pieters, Dimkov, & Pavlovic, 2013)	Text + Venn	+	+	+	+	-	+	-	+	-	+	+	-	+	-	-
(Sarkar, Kohler, Riddle, Ludäscher, & Bishop, 2014)	DAG	-	-	+	+	-	+	-	-	+	-	-	-	+	+	-
(Al Sabbagh & Kowalski, 2012)	UML like	-	-	-	+	-	±	+	+	+	-	-	-	+	-	+
(Pieters, 2011a)	HyperGraph	+	+	+	+	+	-	+	+	+	+	-	-	+	-	-
(Zhao, Huang, Jin, & Zhang, 2011)	Graph	-	-	-	-	-	-	-	-	-	-	-	+	+	+	-
(Kriaa, Bouissou, & Pietre-Cambacedes, 2012)	Graph	-	-	-	+	-	±	-	-	-	-	-	+	+	-	-
(De Nicola, Ferrari, & Pugliese, 1998)	Text	+	+	-	+	-	+	+	-	+	-	-	-	-	-	+
(Vintr, Valis, & Malach, 2012)	Tree	+	±	±	-	-	-	+	-	-	-	-	+	+	±	-
(Santhi, Yan, & Eidenbenz, 2010)	Graphical	-	±	-	+	+	-	+	+	-	-	-	+	+	+	-
(Maiden, Jones, Manning, Greenwood, & Renou, 2004)	UML like	-	-	+	+	-	-	+	+	+	±	±	-	-	-	-
(Franch, 2006)	Graph	-	-	+	+	-	-	+	+	-	-	-	+	-	-	-
(Mohaghegh, Kazemi, & Mosleh, 2009; Mohaghegh, 2010)	Graph	-	-	+	-	-	+	+	+	+	-	-	+	±	±	±

Table 2.1.: The reviewed models and their characteristics. Section 2.1 discusses the characteristics of the TRE$_S$PASS socio-technical security model.

2.2. The TRESPASS Model

We are now ready to discuss the final version of the TRESPASS socio-technical security model. The TRESPASS socio-technical security model advances the state of the art by providing a combination of features that is not supported by of the reviewed models. Major achievements are the naming of components to combine model elements that share the same functionality for the modelled system and the combination of access control policies and processes, which separate authentification from functionality in policy enforcement.

This section follows the same structure as the previews section, discussing representation followed by the individual model components. In Section 2.3 we will then discuss the model's formal underpinnings.

2.2.1. Representation

As discussed before, the TRESPASS model features a hybrid representation approach. Internally, the model is graph based, with the infrastructure on different layers being the main graph. Basing the internal representation on a graph enables the application of many standard, well established analyses. It also is a natural choice of representation related to the navigation metaphor (Pieters et al., 2016) and the attack navigator map (Probst, Willemson, & Pieters, 2016; The TRESPASS Project, D4.2.2, 2016).

The XML format provides the representation for the elements described in the remainder of this section. The schema definitions for the model itself and for the associated scenario are described in Appendix A.

One of the most important features of the model are unique identifiers, which enable the tools to directly address every single part of the model. The unique identifier is used throughout the TRESPASS process and tools to associate properties such as type, quantitative properties, or analysis results with the elements. Identifiers are most often automatically generated by tools, and as such are not expected to be human-readable. Most components expect a second unique attribute such as a name that can be used in the visualisation of the model.

The representation of the model contains sequences of the elements described in the following, together with information about where they are located, if appropriate. This detangling of the model elements and their locations has significantly simplified the handling of the model in the different parts of the TRESPASS process and tools.

The scenario provides the setting for the TRESPASS tools with respect to model, attackers, and goal policy. The attacker can be specified as a variable representing *all* potential actors in the model, or a unique identifier of a specific actor. The goal policies can either address a location, an asset, or an actor (The TRESPASS Project, D1.2.2, 2015).

2.2.2. Infrastructure

The infrastructure layer describes any part of the socio-technical system that provides connectivity between system elements. The infrastructure can provide connectivity at many different layers, for example, the physical layer and the virtual or network layer. Locations in the infrastructure represent places in the organisation, where processes can be located and move.

The *physical* layer is represented as locations in the model. The *virtual* layer is represented as assets, which are described in the next section. Locations in the different layers are connected by edges, which represent the connectivity on the different layers. Edges can also connect elements on *different* layers, for example a computer and the room, where it is located.

The *social* layer is represented by nodes that represent actors or processes. These nodes are special in that they can move around in the modelled system. As described below, this representation eases the handling of assets moving with processes and actors, since they are located at the nodes representing these entities.

Every location has a unique identifier and a name. The name is used in the visualisation. In principle, name and identifier can be the same, wherever appropriate.

2.2.3. Assets and Containment

Assets represent data and items in the model. *Data* has a unique identifier and also has a name, that in principle specifies the data's type or applicability. For example, there may exist several keys opening the same door in a model; each key would have a unique identifier, but would have the same name. The main purpose of this approach is the simplification of expressing policies. Using the name, policies can simply refer to the name of the key; without the keys sharing the name, one would need to replicate the policy for every single identifier. Data can be located at locations, actors, or items. Data can optionally also contain a value.

Items have the same unique identifier and a name, and just as for data, the name is not required to be unique but can be shared by items that have the same applicability. For example, there may exist several ATM cards, that would have unique identifiers but share the same name or type. As for data, this approach is mostly targeted at simplifying the specification of policies. Items can be located at locations, actors, or other items.

The TRESPASS model supports *containment* through the location of data and items: items can be contained in other items, and data can be contained in items. For example, an item representing a hard drive can be contained in a computer, or a file, represented as data, can be contained in an item representing a virtual machine, which in turn is contained in a server.

2.2.4. Processes, Actions, and Behaviour

Processes represent active components in the modeled system, namely processes in the virtual layer and actors in the social layer. In both layers, components can be described through input, output, and movement. The only other action needed is the launching, or execution, of processes; this action represents for example the launching of a process on a computer by an actor.

Processes are represented as sequences of these actions and they have a unique identifier plus a name. Every action takes one or more arguments and a location, where the action should be executed:

- The output actions takes as arguments a sequence of values, variables, or tuples, which in turn can consist of all these elements again;
- The input action takes as arguments a sequence of values, variables, input variables, wildcards, or tuples, which in turn can consist of all these elements again;
- The execute action takes as argument a process that should be executed; and
- The move action takes no arguments, but moves the node, the assets it contains, and the remainder of the process to the target location.

The location for all these actions can be the name of a location, a variable, or the special value “self”, which represents the location, where the process is located.

2.2.5. Actors

Actors are just special cases of processes. Actors have unique identifiers and names, and may “contain” items and data. Like processes, actors can move through the system model; since they are represented as locations, the data and items located at them move with them.

As mentioned above, the TRESPASS socio-technical security model does not distinguish between insiders and outsiders. Instead, different relations to the modeled system can be modeled by giving the actor different amount of access, items, or data.

2.2.6. Policies

Policies play a crucial role in the TRESPASS model (The TRESPASS Project, D1.2.2, 2015): they represent access control policies, they represent the goal to be maintained by an organisation, and they support the separation of the why (credentials) and the what (enabled actions, and in turn enabled processes).

Like all other elements in the TRESPASS model, policies have unique identifiers, but in the case of policies these serve only the purpose of differentiating policies. A policy consists of two components: the required credentials, and the enabled actions. Both components

can be empty or complete — the empty sets represent the cases where nobody is able to execute the action or that no action is enabled, the complete sets represent the case that no credentials are required or that all actions are enabled.

Credentials can be a location, the actor or process must be located at, or data or items with specific values that are required to perform an action. The enabled *actions* can be either any action without arguments, or with arguments like actions in processes as described above. In addition, also output actions may use wildcards to indicate that the policy enables any output that fulfils the non-wildcard elements.

2.2.7. Quantitative Measures

The TRESPASS model supports metrics, or quantitative measures, on all model elements. The model itself does not have a notion of the semantics of values, but only supports storing the values in its representation as keys and values at the elements. The storage and handling of this data is implemented by the TRESPASS knowledge base (The TRESPASS Project, D2.4.1, 2016).

2.2.8. Attacks, Vulnerabilities, and Countermeasures

Just like quantitative measures, attacks, vulnerabilities, and countermeasures are represented in the TRESPASS knowledge base, for example, through adjusting values of some of the elements. Attacks are identified by analysing the scenario and the model it refers to, using for example treemaker (The TRESPASS Project, D3.4.2, 2016) and generating attack trees from it. Vulnerabilities are either chosen as properties of the artefacts in the model when constructing it, or are stored in the knowledge base as result of analysing the identified attacks.

2.3. The TRESPASS Calculus

As described above, the TRESPASS socio-technical security model represents the infrastructure of the socio-technical system to model as nodes in a directed graph (Probst & Hansen, 2008), representing *rooms*, access control points, and similar locations. *Actors* and *processes* are represented by nodes and can possess *assets*, which model data and items that are relevant in the modelled scenario. Assets and other elements in the model can be annotated with a value and a metric, *e.g.*, the likelihood of being lost. Nodes representing assets can be attached to locations or actors; assets attached to actors move around with that actor. Actors and processes perform *actions* on locations, including physical locations or other actors. These actions are restricted by *policies* that represent both access control and the behaviour as expected by an organisation from its employees. Policies consist of required credentials and enabled actions, representing what an actor needs to provide in order to enable the actions in a policy, and what actions are enabled if

an actor provides the required credentials, respectively (The TRESPASS Project, D1.2.2, 2015).

The formalism underlying the TRESPASS modelling approach is based on the Klaim (de Nicola, Ferrari, & Pugliese, 1998) process calculus. In contrast to Klaim, we attach processes and actors to special nodes that move around with the process. This makes the modelling of actors and items carried by actors more intuitive and natural, but can easily be mapped back to original Klaim. The metrics mentioned above can represent any quantitative knowledge about components, for example, likelihood, time, price, impact, or probability distributions. The latter could describe behaviour of actors or timing distributions.

In the following we briefly summarise the formal semantics of our socio-technical models. The calculus follows previous presentations closely and we will therefore not go deep into details here, merely refer to (Probst et al., 2007). As already mentioned, the semantics is based on a variant of the Klaim calculus (de Nicola et al., 1998), called SocTec, which in turn is based on acKlaim (Probst et al., 2007; Probst & Hansen, 2008). The Klaim calculus uses the *tuple space* paradigm, in which systems are composed of a set of distributed nodes that communicate and interact by reading and writing tuples in shared tuples spaces. The following presentation of SocTec is an adaptation and simplification of the calculus presented in (Probst et al., 2007).

2.3.1. Syntax for specifying Socio-Technical Models

In keeping with tradition, the semantics of the SocTec calculus is split into three layers: nets, processes, and actions. Nets define the overall, distributed structure of the system by specifying where individual nodes and tuple spaces are located. Processes and actions define the actual behaviour of the nodes. The syntax of nets, processes, and actions is shown in Figure 2.1. In the SocTec calculus there are two actions for reading a tuple in a remote tuple space: $\mathbf{in}$ for destructive read and $\mathbf{read}$ for non-destructive read. Both these input actions allow for *template* specifications of the tuple(s) to be read, facilitating a simple form of pattern matching with variable binding. The syntax for templates is shown in Figure 2.2 and the corresponding semantics is shown in Figure 2.4.

The syntax of our calculus is shown in Figure 2.1. Since it is very close to Klaim (de Nicola et al., 1998), we first give a very short summary of the main features of Klaim based on (De Nicola, Gorla, & Pugliese, 2005) and then point out the main differences that we introduce here. In Klaim, nets are collections of *nodes* N that may contain *processes* P and *tuple spaces* TS. A tuple space provides a repository of tuples that can be accessed concurrently. A node of N is a pair $l :: C$, where locality $l \in N$ is the name of the node and C is the distributed component located at node l. Components C can be processes or tuples.

Processes P can be built from the inert process $\mathbf{nil}$ and basic actions for read, write, execute, and creation of new nodes. For example, the action $\mathbf{eval}(P)@\ell$ sends process P for execution to node l. The $\mathbf{in}(T)@\ell$ action looks for a matching tuple $\langle et \rangle$ in the TS

ℓ	::=	l	locality	N	::=	$l ::^{\delta} [P]^{\langle n,\kappa_P\rangle}$	single node
	|	u	locality variable		|	$l ::^{\delta} \langle et\rangle$	located tuple
					|	$N_1 \parallel N_2$	net composition
P	::=	$\mathbf{nil}$	null process	a	::=	$\mathbf{out}\,(t)\,@\ell$	output
	|	$a.P$	action prefixing		|	$\mathbf{in}\,(T)\,@\ell$	input
	|	$P_1 \mid P_2$	parallel composition		|	$\mathbf{read}\,(T)\,@\ell$	read
	|	A	process invocation		|	$\mathbf{eval}\,(P)\,@\ell$	migration
					|	$\mathbf{move}\,(\ell)$	move

Figure 2.1.: The syntax of our calculus. We divide localities into those representing infrastructure of any kind (l_i), such as rooms or work stations, and those representing actors (l_a), or more generally processes. We omit the index when both kinds of localities can be used. Tuple spaces contain evaluated tuples. In contrast to Klaim we limit tuples (and templates) to pairs; the first component specifies the name, the second component is either a string (to represent knowledge) or a location (to represent items). Locations have a name (the l) and an access policy δ.

T	::=	$F \mid F,T$	templates	et	::=	$ef \mid ef, et$	evaluated tuple
F	::=	$f \mid !x \mid !u$	template fields	ef	::=	$V \mid l$	evaluated tuple field
t	::=	$f \mid f,t$	tuples	e	::=	$V \mid x \mid \ldots$	expressions
f	::=	$e \mid l \mid u$	tuple fields	ℓ	:=	$l \mid u \mid \mathit{self}$	action target

Figure 2.2.: Syntax for tuples and templates.

located at l. If $\langle et\rangle$ is found, the formal fields of T are replaced in the continuation process with the corresponding names of et. We consider in our adaptation of Klaim the actions $\mathbf{out}, \mathbf{in}, \mathbf{eval}, \mathbf{move}$ for output, input, evaluation of processes, and moving of actors.

Tuples, $\langle t\rangle$ in TS, are sequences of names and are inactive components that have either been there already in the initial configuration or have been output by a process during a computation at this location. The TS located at node l results from the parallel composition of all located tuples positioned at l. Pattern matching is used to select tuples in TS where the syntax $!x$ is used to bind variables to names, for example, *match*$(!x, l)$ produces the substitution $\sigma = [l/x]$. Intuitively, a pattern matches a tuple if both have the same number of fields. Corresponding fields match, if they have the same name, or one is a variable and the other one a name.

One of the key differences between classic Klaim and SocTec is the explicit support for access control policies in the latter, through a *reference monitor* embedded in the semantics. Before going further into the semantics of SocTec, we first need to define these *access control policies*. In the SocTec calculus, the kind of access that is relevant to control, is

whether or not a process at a given location is allowed to perform a specific action at a remote location. Thus we can formalise access control policies as follows:

$$\begin{aligned} \pi \subseteq \mathsf{AccMode} &= \{\mathbf{i}(template)@location, \mathbf{r}(template)@location, \mathbf{o}(tuple)@location, \\ &\quad \mathbf{e}(process)@location, \mathbf{m}(location))\} \\ \delta \in \mathsf{Policy} &= (\mathsf{Loc} \cup \{\star\}) \to \mathcal{P}(\mathsf{AccMode}) \end{aligned}$$

where the *access modes* correspond to the actions that can be taken in the semantics: $\mathbf{i}$ for (destructively) reading a tuple, $\mathbf{r}$ for (non-destructively) reading a tuple, $\mathbf{o}$ for outputting (writing) a tuple, $\mathbf{e}$ for remote evaluation of a process, and $\mathbf{m}$ for moving an actor or process node to another location. The special '$\star$' location is used to denote default policies, i.e., access modes that are allowed from all locations not specifically mentioned. Templates in the policy specification stand for a possibly empty sequence of values, variables, or input variables from Figure 2.2, tuples stand for a possibly empty sequence of values and variables, and location may be empty or be a location value, a location variable, or the special value *self*, that stands for the location where the action is executed.

Data in the SocTec calculus is also restricted compared to original Klaim; we only allow tuples of length two, where the first element is the name of the tuple and the second element contains its value, which can be a string or a location. Items that are known by actors are represented by pairs of strings; items that can contain further information are represented as pairs of a string and a location. Templates and tuples used in actions, on the other hand, can contain more elements, for example, for modelling IP traffic on the network. However, tuples for these modelling artefacts are not supposed to be part of the model.

2.3.2. Semantics of Socio-Technical Models

We can now introduce the semantics for SocTec, by defining the reduction relation for processes and actions, shown in Figure 2.3. As described above, processes are in principle composed of sequences of actions, (sub-)processes that execute in parallel, or a recursive invocation through a place-holder variable. The actions a process can perform are: $\mathbf{out}$, that writes a tuple to the specified tuple space; $\mathbf{in}$, that reads a tuple (at the specified tuple space) matching the template and then *removes* the tuple in question; $\mathbf{read}$ that also reads a tuple (at the specified tuple space) matching the given template but does *not* remove the tuple; $\mathbf{eval}$ that evaluates the given process at the specified (remote) location. Finally, the $\mathbf{move}$ action relocates the node representing the actor or process.

For the specification of socio-technical security models, however, we only wish to allow certain kinds of processes and certain moves between nodes, e.g., a node representing a (physical) actor should only be able to move between nodes representing physical localities. For the specification of processes we only allow sequential sequences of actions without parallel composition. We need this constructor for the simulation of processes or models to identify, for example, actions that have been performed in parallel (Bæk & Bach, 2016).

The restriction on moves is formalised in the form of the so-called *infrastructure* of the underlying nets:

$$\mathcal{I} \in \mathsf{Infrastructure} = \mathcal{P}\left(\mathsf{Locality} \times \mathsf{Locality}\right)$$

Essentially, the infrastructure represents the graph discussed in Section 2.2, relating the pairs of nodes between which moves are allowed (still subject to access control rules).

In addition to the reduction relation, the semantics also incorporates a structural congruence, simplifying (re-)presentation of, computation with, and reasoning about processes and nets. The congruence is shown in Figure 2.5.

$$\frac{[\![t]\!] = et \qquad \boxed{(l,l') \in \mathcal{I} \wedge \mathbf{o} \in \delta(l')}}{l ::^{\delta} [\mathbf{out}\,(t)\,@l'.P]^{\langle n,\kappa_{\mathbf{out}(t)@l'.P}\rangle} \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle} \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P]^{\langle n,\kappa_P\rangle} \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle} \parallel l' ::^{\delta'} \langle et\rangle}$$

$$\frac{match([\![T]\!], et) = \sigma \qquad \boxed{(l,l') \in \mathcal{I} \wedge \mathbf{i} \in \delta(l')}}{l ::^{\delta} [\mathbf{in}\,(T)\,@l'.P]^{\langle n,\kappa_{\mathbf{in}(T)@l'.P}\rangle} \parallel l' ::^{\delta'} \langle et\rangle \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P\sigma]^{\langle n,\kappa_{P\sigma}\rangle} \parallel l' ::^{\delta'} \mathbf{nil}}$$

$$\frac{match([\![T]\!], et) = \sigma \qquad \boxed{(l,l') \in \mathcal{I} \wedge \mathbf{r} \in \delta(l')}}{l ::^{\delta} [\mathbf{read}\,(T)\,@l'.P]^{\langle n,\kappa_{\mathbf{read}(T)@l'.P}\rangle} \parallel l' ::^{\delta'} \langle et\rangle \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P\sigma]^{\langle n,\kappa_{P\sigma}\rangle} \parallel l' ::^{\delta'} \langle et\rangle}$$

$$\frac{\boxed{(l,l') \in \mathcal{I} \wedge \mathbf{e} \in \delta(l')}}{l ::^{\delta} [\mathbf{eval}\,(Q)\,@l'.P]^{\langle n,\kappa_{\mathbf{eval}(Q)@l'.P}\rangle} \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle} \rightarrowtail_{\mathcal{I}} l ::^{\delta} [P]^{\langle n,\kappa_P\rangle} \parallel l' ::^{\delta'} [Q]^{\langle n,\kappa_Q\rangle} \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle}}$$

$$\frac{\boxed{\{(l,l''),(l'',l')\} \in \mathcal{I} \wedge \mathbf{m} \in \delta(l')}}{l ::^{\delta} [\mathbf{move}\,(l')\,.P]^{\langle n,\kappa_{\mathbf{move}(l').P}\rangle} \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle} \rightarrowtail_{\mathcal{I}'} l ::^{\delta} P \parallel l' ::^{\delta'} [P']^{\langle n,\kappa_{P'}\rangle}} \quad \mathcal{I}' = \mathcal{I} \setminus (l,\_) \cup \{(l,l')\}$$

$$\frac{L \vdash N_1 \rightarrowtail_{\mathcal{I}} L' \vdash N_1'}{L \vdash N_1 \parallel N_2 \rightarrowtail_{\mathcal{I}} L' \vdash N_1' \parallel N_2} \qquad \frac{N \equiv N_1 \qquad L \vdash N_1 \rightarrowtail_{\mathcal{I}} L' \vdash N_2 \qquad N_2 \equiv N'}{L \vdash N \rightarrowtail_{\mathcal{I}} L' \vdash N'}$$

Figure 2.3.: Reduction semantics for SocTec.

$$match(V,V) = \epsilon \quad match(!x,V) = [V/x] \quad match(l,l) = \epsilon \quad match(!u,l') = [l'/u]$$

$$\frac{match(F,ef) = \sigma_1 \qquad match(T,et) = \sigma_2}{match((F,T),(ef,et)) = \sigma_1 \circ \sigma_2}$$

Figure 2.4.: Semantics for template matching.

$$N_1 \parallel N_2 \equiv N_2 \parallel N_1 \qquad (N_1 \parallel N_2) \parallel N_3 \equiv N_1 \parallel (N_2 \parallel N_3)$$

$$l ::^{\delta} [P]^{\langle n,\kappa_P\rangle} \equiv l ::^{\delta} [(P \mid \mathbf{nil})]^{\langle n,\kappa_{(P \mid \mathbf{nil})}\rangle} \qquad l ::^{\delta} [A]^{\langle n,\kappa_A\rangle} \equiv l ::^{\delta} [P]^{\langle n,\kappa_P\rangle} \quad \text{if } A \triangleq P$$

$$l ::^{\delta} [(P_1 \mid P_2)]^{\langle n,\kappa_{(P_1 \mid P_2)}\rangle} \equiv l ::^{\delta} [P_1]^{\langle n,\kappa_{P_1}\rangle} \parallel l ::^{\delta} [P_2]^{\langle n,\kappa_{P_2}\rangle}$$

Figure 2.5.: Structural congruence on nets and processes.

3. Other Models Explored in TRESPASS

In this chapter we discuss some alternative models we have explored in TRESPASS to address specific questions or to explore extensibility of models (The TRESPASS Project, D1.3.2, 2015). These alternative approaches explore how to add time to the model (Section 3.1), how to extend the model with new actions and properties, and to perform probabilistic analyses (Section 3.2), how to model the flow of money and processes (Section 3.3), and finally, how to interface to established tool chains (Section 3.4).

The models discussed here contribute to the TRESPASS socio-technical security model by illustrating extensions in the style of (The TRESPASS Project, D1.3.2, 2015), be it on the action or the modelling level, or be it as an interface between our models and an industrial tool chain.

3.1. Timed Automata as Models

The TRESPASS socio-technical security model does not *directly* support time; this is however an important property of attacks, for example to assess the time needed for an attack (N. David, 2014) or to judge where an attacker may be either while launching the attack or after having performed it.

Timed automata are one way of directly providing time as a first-class citizen in models, by providing formal semantics and modelling for distributed systems with real-time constraints (Alur & Dill, 1994). A timed automaton extends the classic notion of finite automaton with real-valued clocks and transitions conditioned on values computed over the set of clocks. Timed automata have since been extended with numerous features, for example, *priced timed-automata* and *probabilistic timed automata* specifically designed to handle systems with (quantitative) cost and probabilistic behaviour respectively. Furthermore, timed automata have also been used in the context of *timed games* in which system behaviour can be modelled as a two player game with time constraints.

Translating the socio-technical security model to a network of timed automata has the added benefit of making available the extensive tool support for modelling, analysis, and verification of systems using timed automata, including UPPAAL[1], a state of the art model checker for networks of timed automata (Behrmann, David, & Larsen, 2004). Tool support is also available for many of the extensions of timed-automata, such as UPPAAL TIGA for model checking of safety and reachability properties in *timed games*, UPPAAL SMC

[1] http://uppaal.org/

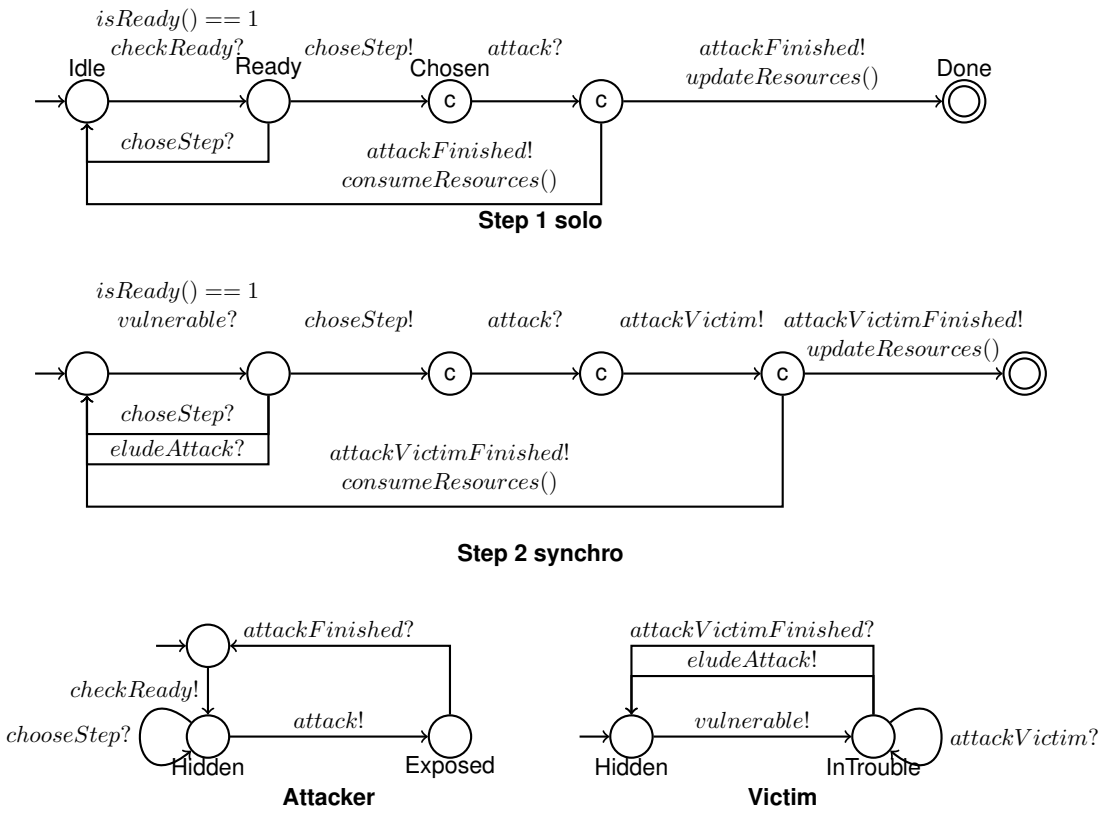

Figure 3.1.: Timed automata model of a (generic) two-step attack

for *statistical model checking* (A. David, Larsen, Legay, Mikucionis, & Wang, 2011) which supports statistical model checking of hybrid models.

In the TRESPASS project, we have investigated timed automata as a formalism for identifying and modelling attacks that complements and extends the capabilities of attack trees (N. David, 2014; N. David et al., 2015; Gadyatskaya et al., 2016). The main attraction of using timed automata in this scenario is the possibility to translate the model into a network of timed automata that then in turn model the attack. In particular, it is straightforward to model complex behaviour as well as including several actors, e.g., both attacker(s) and victim(s) with different capabilities and skill sets, while retaining strong tool-support. Figure 3.1 shows a generic timed automata model of attacks composed of two steps: one step that the attacker can take regardless of the actions of the victim (the topmost automaton) and one that depends on actions taken by the victim, e.g., leaving home or (not) locking the front door. The lower two automata model the overall behaviour of attacker and victim respectively.

Even in this simple example, a number of important properties of the timed automata approach to modelling attacks can be seen. First of all, the model includes, and unifies, both the attacker actions and victim actions. Thus it is easy to model both defensive measures taken by the victim, but also complex behaviour of the victim that may either increase or

decrease the degree of vulnerability of the victim to a particular attack. Second, it is trivial to add more victims and attackers (with different individual behaviour) directly in the model, allowing for more complex attacks based on attacker/victim interaction to emerge from the model. Third, although not visible in the figure, timing constraints on both the behaviour of the attacker and the victim are trivially incorporated into the model, allowing for even more precise models of an attack and/or of the cost profile of the attack. In addition to timing constrains other quantitative constraints, e.g., cost, may be added to the model in a straightforward way. Finally, both the basic and the extended models enjoy strong automated tool support, including model checking and statistical simulation.

Further details on the semantics and statistical model checking can be found in (The TRE$_S$PASS Project, D3.2.1, 2015; Behrmann et al., 2004).

3.2. The STS Model

The socio-technical system model (Lenzini, Mauw, & Ouchani, 2015; The TRE$_S$PASS Project, D3.4.1, 2014) is another approach investigated in the course of the TRE$_S$PASS project. The STS formalism models a subset of the TRE$_S$PASS socio-technical security model: it models a system's physical infrastructure, physical assets present in the building, and containers of objects. Additionally, the actions of human agents can be modeled, including their movement and physical interaction with objects. Actions in STS have a cost and the different choices that agents take are guided by probabilities or by contextual conditions.

The semantics of the STS model is specified as a transition system that captures important socio-technical security properties, e.g., expressing the possibility, the likelihood, and the cost of actions such as intrusions and theft by an malicious intruder.

For the analysis, STS models are mapped to probabilistic models that can be analysed by the PRISM model checker with respect to security requirements expressed in PCTL.

The investigation of STS has provided important insights to the TRE$_S$PASS project: it has explored the *extension of models with new actions*, namely the locking and unlocking of doors and destroying objects, the *extension of objects with states* (The TRE$_S$PASS Project, D1.3.2, 2015), and the probabilistic modeling and analysis using PRISM. The main insight of the work with STS is the trade-off in socio-technical language expressiveness and the investment required to perform security assessment with this language.

An STS model consists of a tuple $\langle Phy, Obj, Act, Int, Struc \rangle$, where Phy models the physical space, Obj models objects, Act the actors, Int the intruder. $Struc$ is a hierarchical structure that describes the relations that exist between the STS's entities, such as locations in a building, doors between locations, objects that are contained in other objects. The infrastructure Phy models locations, doors, and relates keys to doors and doors to their state, namely locked or unlocked. An object can be a container, *e.g.*, a box or a safe, and containers are lockable like doors. Objects can be moved and destroyed. Actors

have a behaviour, which is represented as a sequence of basic actions, chosen non-deterministically, conditionally, or probabilistically. Executing an action has a cost. The available actions move an actor from one location to another, lock or unlock a door or a container, pickup or put down an object from a location or a container, destroy an object, or give or receive an object from another actor.

An actor's behaviour is built up starting from the basic actions and from a special action $Stop$, expressing inaction. Composite actions are built by sequential composition, non-deterministic choice, probabilistic choice, and guarded choice. The guard is a contextual condition that checks for example whether an agent possesses the right key, whether a door is unlocked, whether an object is movable, and so on.

Finally, an intruder can also possess objects and can act in any possible way that is in relation to what the infrastructure and the objects allow. The intruder is modelled as a regular actor, but with fewer restrictions, resulting in additional power. Intruders can execute any possible action, so the probability of an intruder action is always 1. The intruder can also open locked objects and doors without needing keys, and can stealthily place objects at actors or other objects. Finally, an intruder can impersonate anyone in the exchange of objects with others. There are however limitations to the intruder's abilities: an intruder cannot teleport, cannot walk through walls, and must move step by step. An intruder cannot get or put objects from distance, that is intruders must be in the same location as their victim. Intruders can also not change the physical properties of objects: the intruder cannot move an unmovable object, or destroy an indestructible object.

Changes to the state of the STS are performed by a labelled transition system that, starting from an initial state performs transitions in the set of all possible STS states. The transition relation $\Rightarrow$ defines how STS states change in consequence of what the actors and the intruder do. It is the smallest relation that satisfies the transition rules (Lenzini et al., 2015).

An overview of the security analysis performed with an STS model is depicted in Figure 3.2. An STS model is mapped to a Probabilistic Symbolic Model Checker (PRISM) program (Lenzini et al., 2015). The figure also shows that, alongside, security template expressions that define relevant security properties are instantiated as extended Probabilistic Computation Temporal Logic (PCTL) formulas. PRISM uses this input to check the satisfiability of security properties in the considered model, and produces the checking result in terms of probability and cost.

3.3. Value and Process Models

The Telecommunication Services Case Study and the Cloud Case study present special challenges with regard to Risk Assessment. Information relevant for identifying, understanding and analysing some of their specific risks is not always available in the TRE$_S$-PASS model, for example, the money flows of the Telecom fraud scenarios and the cloud administration processes of the Cloud scenarios.

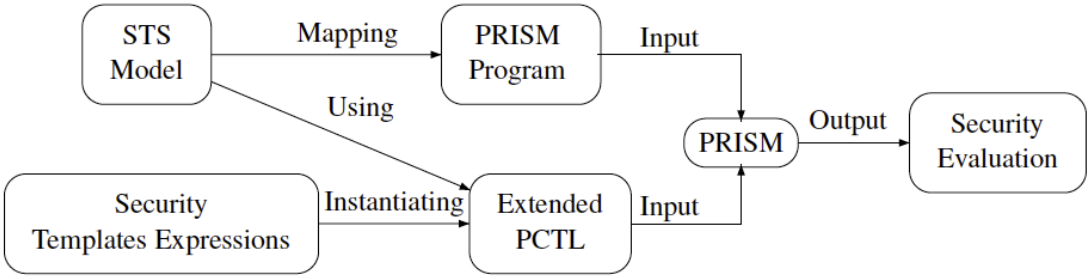

Figure 3.2.: Scheme resuming our approach for a socio-technical security evaluation

Value models. Business value models are geared towards representing value transfers between profit-loss responsible actors. Because of this, they are a natural choice for modelling money flows. Thus, *e3value*, one of the most prominent value models was chosen and applied to the Telecom case study.

Overall, we have observed that e3value models better captured economic transactions of the two misuse scenarios modelled, compared to the TRESPASS model. Furthermore, e3value was somewhat easier to create as less knowledge about the technical infrastructure was needed and the building blocks and syntax was simpler. In this sense, it not only overcame several of the TRESPASS (WP1) language constraints but also provided more flexibility and ease of use when discussing telco fraud.

However, we do expect that for similar attacks, which (also) exploit technical or social vulnerabilities, e3value models will not be sufficient. Such technical vulnerabilities may be exploited in scenarios involving hacking of a TSP's or TSP customer's infrastructure (e.g. CPE - Customer-premises equipment, PBX - Private Branch Exchange). For such scenarios, both the value model and the TRESPASS model are necessary.

Process models. In cloud environments, the architecture is defined by the virtual infrastructure just as much as by the physical infrastructure. However, unlike the physical infrastructure, the virtual one is highly dynamic. Besides the resource management aspect, which is mostly taken care of by the Hypervisor software, there are also a number of administrative processes, such as deploying a new Virtual Machine. Many of the cloud-specific Risks exploit weaknesses of one of these processes. As such, an assumption was made that a thorough risk Assessment of a cloud infrastructure should also look at how these processes are specified and implemented. However, there is not much research in assessing the risks associated with a process.

3.4. Extensions in the ArchiMate Language

Finally, we have investigated the mapping between the ArchiMate enterprise architecture modelling standard and the TRESPASS socio-technical security model (The TRESPASS

Project, D1.3.2, 2015). The ArchiMate standard (The Open Group, 2013) describes two main mechanisms that organisations or end-users can apply to extend the language: (1) by means of *specialisation* of language concepts (elements or relationships), similar to the use of stereotypes in UML, and (2) by means of adding (sets of) *attributes* to language concepts. Interfacing to the ArchiMate standard is especially interesting since it makes a wide selection of industrially used tools for modelling and assessing risk and security aspects available.

One approach suggested by The Open Group is a "risk and security overlay" for the ArchiMate language, defining specialisations of both ArchiMate core elements and elements from the Motivation extension (the latter mainly for risk analysis). Figure 3.3 shows the structure of this overlay. The red elements are specialisations of core and motivation elements to model the results of a risk assessment, while the green elements are specialisations of motivation elements to model control objectives, principles and control measures.

The ArchiMate language, extended with the risk and security overlay as described in Section 3.4, has been applied in the IPTV case study to model the infrastructure using standard ArchiMate core elements (see Figure 3.4). This infrastructure model makes the TRE$_S$PASS model easily accessible to modellers already familiar with ArchiMate, and it can serve as a basis for vulnerability and risk assessment. The red ellipses in Figure 3.4 indicate potential vulnerabilities in the IPTV setup, which may be of a technical nature (e.g., the bluetooth connection could be susceptible to eavesdropping), but also of a social nature: the secondary cardholder has access to the same payment functions as the account holder, which is a risk in case of an untrustworthy secondary card holder.

Figure 3.5 shows the details of a risk assessment of the latter vulnerability, a potentially untrustworthy secondary card holder that has access to the payment service. This model uses specialised ArchiMate elements from the risk and security overlay. A possible threat event based on this vulnerability is abuse of access to the payment services, by making

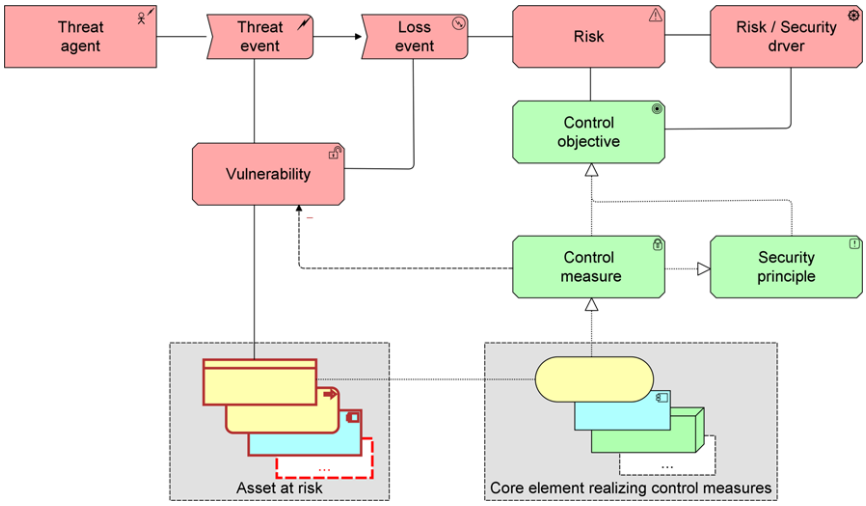

Figure 3.3.: ArchiMate risk and security overlay

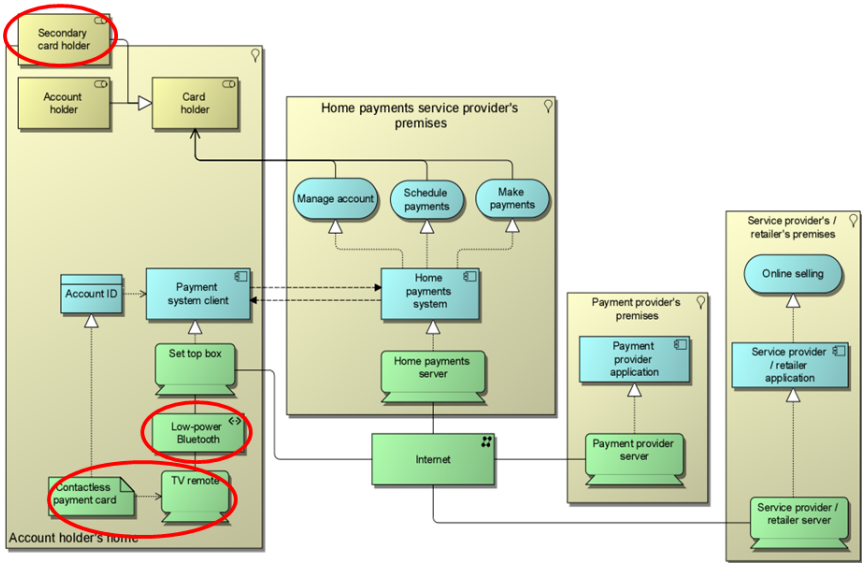

Figure 3.4.: IPTV infrastructure modelled in the ArchiMate language

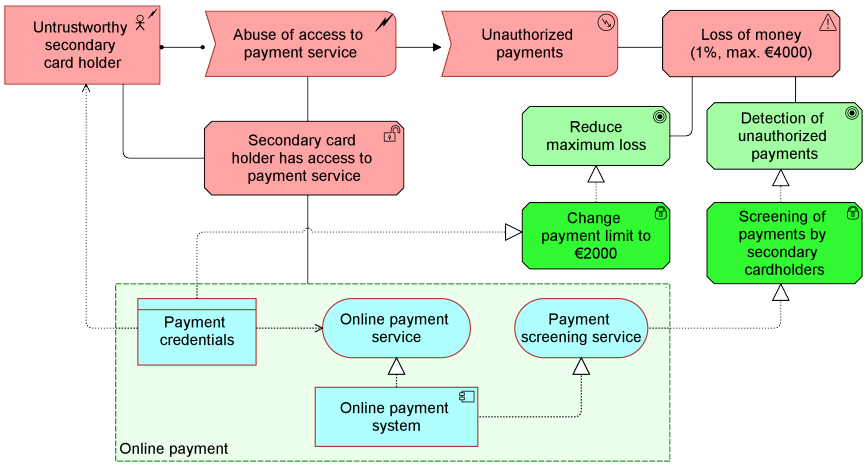

Figure 3.5.: Example ArchiMate risk analysis model for the IPTV case

unauthorised payments (e.g., to the personal account of the secondary card holder). The risk element provides a quantification of this, in terms of likelihood or frequency (abuse is estimated to occur in 1% of the cases) and probable loss magnitude.

4. Modelling Socio-technical Systems

We will now illustrate the use of the TRE$_S$PASS socio-technical security model by applying it to parts of the case studies investigated in the project. The different cases highlight and present different examples for how to model elements of socio-technical systems.

As discussed above, the TRE$_S$PASS socio-technical security model is represented as XML, and we will discuss the model representation both in this archaic form and in a more abstract form. The schema definitions for the XML format of the model and the scenario are included in Appendix A.1 and Appendix A.2, respectively, and also available online (The TRE$_S$PASS Project, 2016b, 2016c). The XML files for the models discussed here are included in Appendix B and also available online (The TRE$_S$PASS Project, 2016a).

For the IPTV case we will show snippets of the XML representation of the model. Since the XML files for all cases are available from the appendix and online, we only show XML fragments to illustrate noteworthy points.

4.1. IPTV

In this section we describe the "IPTV case study" considered in the TRE$_S$PASS project.[1] The technical details as well as the people and companies involved are confidential and we therefore present an anonymised and slightly redacted version of the original case study. However, all the important features have been retained, and the processes are fundamentally the same as in the original case study.

The case study concerns a system supporting primarily elderly and disabled people in performing online payments and managing their own money from their home. With the target demographic in mind, the system should be integrated into an existing device that is familiar and easy to use for the intended user groups, namely the television set. In practice this is accomplished by hooking up a small, dedicated computer to the TV and an enhanced remote control with a built-in card reader for authentication.

This case study offers many different security aspects to consider: from the strictly technical, such as how information is protected while stored or transmitted, to the socio-technical, covering security issues arising from the use of and interaction with the technology. Within the project, we have explored the socio-technical features of the case study, as a means of validating the methods and tools which are being developed. In particular

[1] Here IPTV refers to television service(s) over the IP protocol.

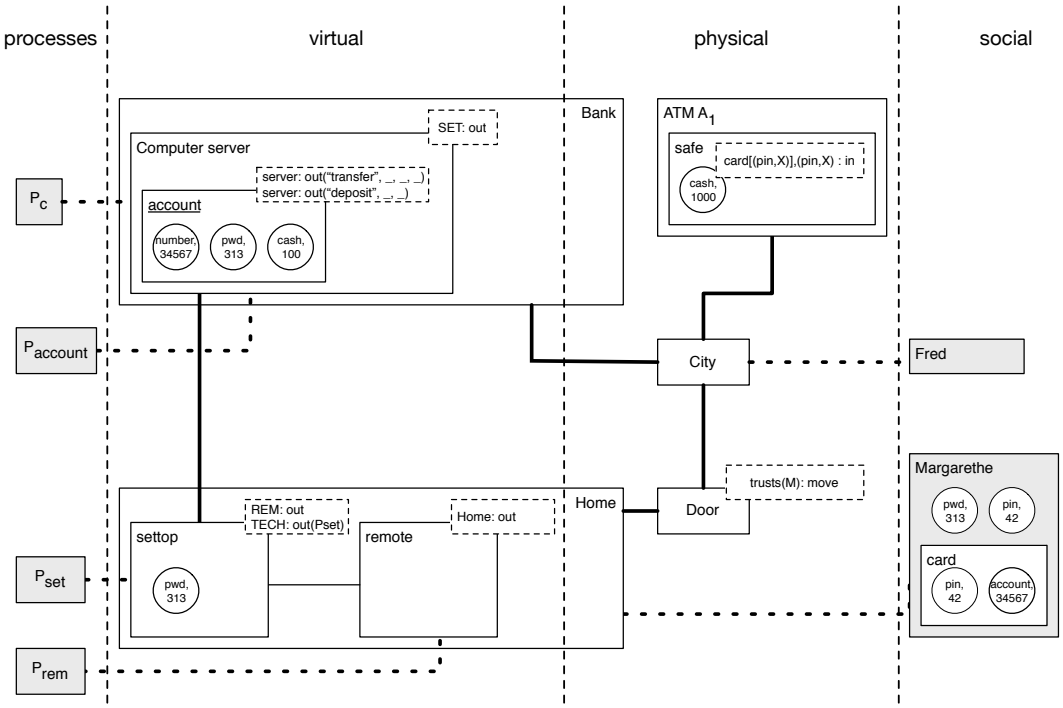

Figure 4.1.: A model for the IPTV case study, consisting of the social layer (actors), the physical layer (infrastructure), and the virtual layer (computer network and processes). The **gray boxes** represent processes, that are discussed in the text below, or actors, and the **outermost white boxes** represent physical locations. **Contained white boxes** represent items, such as cards, and **white circles** represent data. The containment of boxes and circles represents location or containment of the data and items they represent. The dashed boxes represent policies. Dotted lines connect actor and process locations with their current location in the model, and solid lines represent directed or undirected connections between locations.

this case study has provided a context in which to explore the many possible approaches to handling social data.

In the IPTV case study there are two primary actors; an attacker and a victim (the IPTV owner/user). Under normal operation, the user would use a password and a card to interact with the IPTV to use different services, e.g., pay a bill or transfer money, by using a payment card with concomitant PIN code. The payment card is read by the IPTV remote control on which the PIN code is also entered.

The design is intended to be simple and offer the means for people of all ages and abilities to be able to access the services they require. It is intended to complement other means of delivery, not to replace them. In particular, it offers the opportunity for people who are not familiar or comfortable with mobile technology to receive the benefits of the ever-increasing range of mobile services in their homes via their television screen (Egelman,

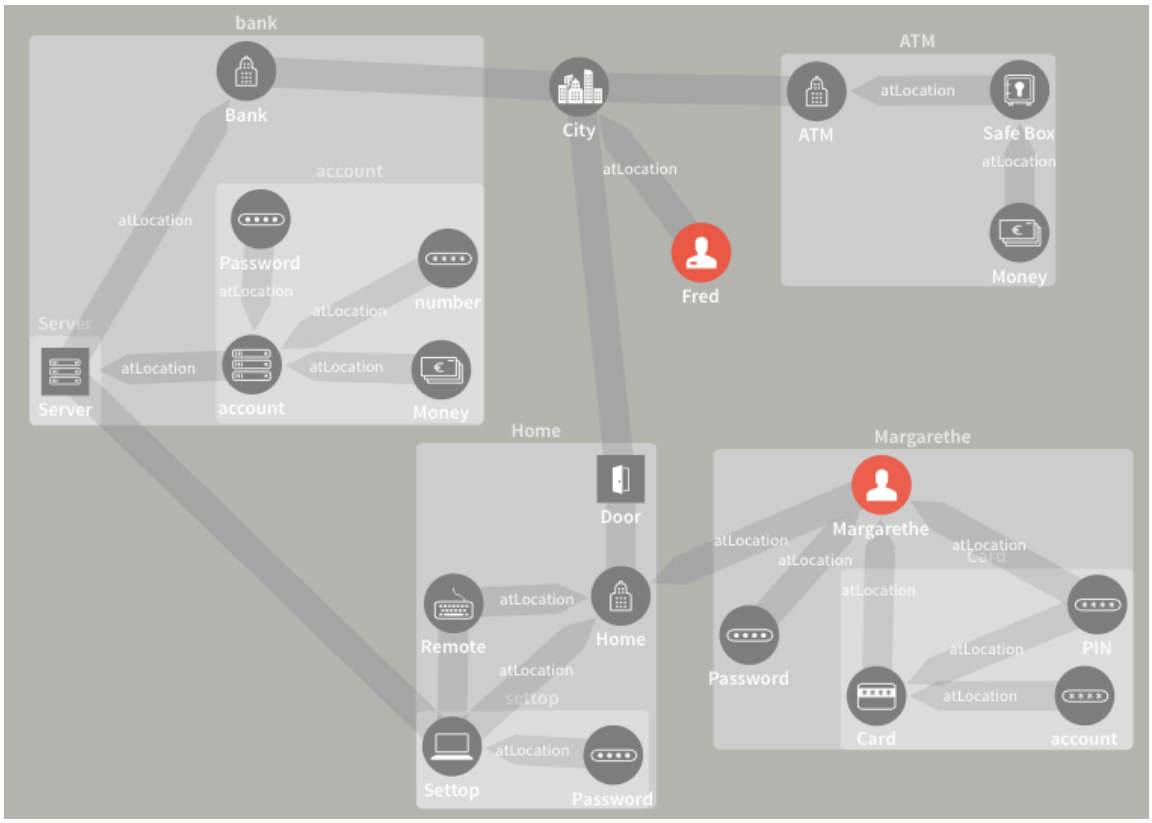

Figure 4.2.: The Attack Navigator Map for the IPTV model from Figure 4.1.

Brush, & Inkpen, 2008). Although this system in principle offers great convenience, it also has the potential to expose the account holder to significant social risks, particularly those stemming from the involvement of both professional carers and family members. These carers could be considered as knowledge insiders, with the potential to act as malicious insiders.

4.1.1. The Model

Figure 4.1 shows one model for the IPTV case, and Figure 4.2 shows the Attack Navigator Map for this model. To illustrate the different layers of the model, the figure makes explicit the virtual layer and processes, the physical layer, and the social layer.

4.1.2. Social Layer

On the social layer, the actors in the system are Margarethe, Charlie, Fritz, and Fred. Margarethe is located at her home, and Charlie, Fritz, and Fred might be in the city.

The actors in the system are represented in the XML file as

```
<actors>
    <actor id="Fred" name="Fred"><atLocations>city</atLocations></actor>
    <actor id="Margarethe" name="Margarethe"><atLocations>home</atLocations></actor>
    <actor id="Charlie" name="Charlie"><atLocations>city</atLocations></actor>
    <actor id="Fritz" name="Fritz"><atLocations>home</atLocations></actor>
</actors>
```

Note that none of the objects refers to what is located at it; the model specification only contains information on where the element in question is located.

4.1.3. Physical Layer

On the physical layer there exist a number of items:

- The ATM has a safe that contains some money. The policy at the safe requires a user to provide a card containing a pin and a matching pin to input some money from the safe. The requirement that the provided pin and the pin on the card match is expressed by the variable *X* in the policy;
- The bank contains a computer C and the home the settop box *SET*, representing the connection between the physical and the virtual layer;
- The door to Margarethe's home has a policy that only actors trusted by Margarethe may enter. Predicates can be provided with arbitrary values, in the example system the IPTV technician might be trusted, as might be family members of Margarethe; and
- Margarethe, has an ATM card, that stores her pin and her account number; the second user, Fred, is not assumed to have any knowledge of the system and its components.

The XML file contains a list of locations:

```
<locations>
  <location id="door"/>
  <location id="home"/>
  <location id="city"/>
  <location id="bank"/>
  <location id="atm"/>
</locations>
```

It is important to name the decision for making something an item, an actor, a process location, or a "real" location. All these locations can contain items and data, but the connection mechanisms are slightly different. Locations can be connected to each other. Processes and actors can only be located at other locations (the "locatedAt" flag at the item in question). Item and data, however, can be located at other items or at locations, and they can be connected by network edges, if relevant.

The XML file contains a number of item and data definitions. It may sometimes be difficult to decide whether to model something as data or an item; a first help is the questions whether there should be a process at the asset; if yes, it should be an item.

```
<data id="owner" name="Owner" value="Margarethe"><atLocations>card</atLocations></data>
<data id="pin" name="PIN" value="1"><atLocations>Margarethe card</atLocations></data>
<data id="password" name="Password" value="2"><atLocations>Margarethe settop account</atLocations></data>
<data id="account" name="account" value="1234"><atLocations>card</atLocations></data>
<data id="number" name="number" value="1234"><atLocations>account</atLocations></data>
<item id="money" name="Money"><atLocations>safe-box account</atLocations></item>
<item id="card" name="Card"><atLocations>Margarethe</atLocations></item>
<item id="safe-box" name="SafeBox"><atLocations>atm</atLocations></item>
<item id="account" name="account"><atLocations>server</atLocations></item>
```

4.1.4. Virtual Layer

Moving to the virtual layer, the model contains the networked components:

```
<item id="server" name="Server"><atLocations>bank</atLocations></item>
<item id="settop" name="Settop"><atLocations>home</atLocations></item>
<item id="remote" name="Remote"><atLocations>home</atLocations></item>
```

Beyond these networked components, the virtual layer only contains direct connections between the networked components. In the next section we will discuss the design space for processes, for example, how we model IP traffic and routers.

4.1.5. Policies

At the remote control, Margarethe is allowed to output assets; the remote control is connected to the settop box, and is allowed to output there, which in turn is allowed to output at the computer at the bank. At the door, an actor must be trusted to answer, and at the safe-box a card and a matching pin are needed to input money. At the account, finally the compute is output commands to withdraw or deposit money into his account:

```
<policies>
  <policy id="p001">
    <credentials>
      <credPredicate name="trusts"><value>Margarethe</value><variable>X</variable></credPredicate>
    </credentials>
    <enabled>
      <move />
    </enabled>
    <atLocations>door</atLocations>
  </policy>
  <policy id="p002">
    <credentials>
      <credItem name="Card"><credData name="PIN"><variable>Y</variable></credData></credItem>
      <credData name="PIN"><variable>Y</variable></credData>
    </credentials>
    <enabled>
      <in />
    </enabled>
    <atLocations>safe-box</atLocations>
  </policy>
  <policy id="p003">
    <credentials>
      <credLocation name="home"/>
    </credentials>
    <enabled><out /></enabled>
    <atLocations>remote</atLocations>
  </policy>
  <policy id="p004">
    <credentials>
      <credLocation name="Remote"/>
    </credentials>
    <enabled><out /></enabled>
    <atLocations>remote</atLocations>
  </policy>
  <policy id="p005">
    <credentials>
      <credPredicate name="technician"><variable>X</variable></credPredicate>
    </credentials>
```

```
    <enabled><out><value>Firmware</value><wildcard /></out></enabled>
    <atLocations>settop</atLocations>
  </policy>
  <policy id="p006">
    <credentials>
      <credLocation name="server" />
    </credentials>
    <enabled>
      <out><value>transfer</value><wildcard /><wildcard /><wildcard /></out>
    </enabled>
    <atLocations>account</atLocations>
  </policy>
  <policy id="p007">
    <credentials>
      <credLocation name="server" />
    </credentials>
    <enabled>
      <out><value>deposit</value><wildcard /><wildcard /></out>
    </enabled>
    <atLocations>account</atLocations>
  </policy>
  <policy id="p008">
    <credentials>
      <credLocation name="settop"/>
    </credentials>
    <enabled><out /></enabled>
    <atLocations>server</atLocations>
  </policy>
 </policies>
```

4.1.6. Processes

The process at the remote control reads the card, the pin, the settop box password, the amount to transfer, and the account to transfer to, then checks the pin against the pin on the card, and sends a transfer request to the settop box:

$$\begin{aligned} P_{remote} \quad := \quad & \mathbf{in}(\texttt{'card'}, !card).\mathbf{in}(\texttt{'pin'}, !userpin).\mathbf{in}(\texttt{'password'}, !pwd). \\ & \mathbf{in}(\texttt{'amount'}, !amount).\mathbf{in}(\texttt{'account'}, !account). \\ & \mathbf{in}(\texttt{'pin'}, userpin)@card.\mathbf{in}(\texttt{'account'}, !ownaccount)@card. \\ & \mathbf{out}(\texttt{'transfer'}, pwd, amount, ownaccount, account)@\texttt{settop} \end{aligned}$$

The matching pin is tested by inputing the tuple $(\texttt{'pin'}, userpin)$ from the card, where *userpin* is the value input by the user. If the user has input the correct pin, this action succeeds and the process continues, otherwise it blocks. The tuple sent to the settop box contains a command (`'transfer'`), the password, the amount to transfer, the user's account and the target account.

At the settop box, a similar process waits for the transfer command, then checks the password against the one stored in the settop box, and if this is correct, outputs a tuple at the bank computer:

$$\begin{aligned} P_{settop} \quad := \quad & \mathbf{in}(\texttt{'transfer'}, !pwd, !amount, !ownaccount, !account). \\ & \mathbf{in}(\texttt{'password'}, \texttt{'pwd'}). \\ & \mathbf{out}(\texttt{'transfer'}, pwd, amount, ownaccount, account)@\texttt{server} \end{aligned}$$

As before, the matching of the password is checked by inputing the tuple $(\texttt{'password'}, pwd)$ at the settop box; if the provided password is correct, this succeeds and the process continues, otherwise it blocks.

At the computer $server$, a process inputs the tuple and outputs the transfer to the account location. Note that we assume that the password for the settop box and the account are the same; they could easily be different, or the account could not require a password (or could implement a challenge/response protocol):

$$\begin{aligned} P_{server} \quad := \quad & \mathbf{in}(\texttt{'transfer'}, !pwd, !amount, !ownaccount, !account). \\ & \mathbf{out}(\texttt{'transfer'}, account, pwd, amount)@ownaccount \end{aligned}$$

Finally, at the account, the password is checked again, and the balance is input and then output again. The latter simulates the checking of the balance; arithmetic and checking are not part of the calculus, but could easily be added:

$$\begin{aligned} P_{account} \quad := \quad & \mathbf{in}(\texttt{'transfer'}, !account, !pwd, !amount). \\ & \mathbf{in}(\texttt{'password'}, \texttt{'pwd'}).\mathbf{in}(\texttt{'cash'}, !cash).\mathbf{out}(\texttt{'cash'}, cash) \end{aligned}$$

As touched upon above, this is only one way of modelling the functionality of transferring money via the settop box, checking the pin and password, and so on. The possible attacks that can be identified depend on the way the system is modelled, and so do the detail of the identified attacks. For example, we will discuss in the ATM case study in Section 4.4, how the keyboard and card reader at the ATM can be modelled, and how this enables the detection of typical attacks on the ATM.

Another abstraction in this case is the modelling of the communication between the network components. In a realistic model, the communication should use IP traffic. This is discussed in the cloud case in Section 4.3.

In the XML file, the abstract notion of processes shown above is mapped into tags:

```
<processes>
    <process id="P_remote">
        <atLocations>remote</atLocations>
        <actions>
            <in><value>card</value><input>card</input></in>
            <in><value>pin</value><input>userpin</input></in>
            <in><value>password</value><input>pwd</input></in>
            <in><value>amount</value><input>amount</input></in>
            <in><value>account</value><input>account</input></in>
            <in><value>pin</value><variable>userpin</variable><locvar>card</locvar></in>
            <in><value>account</value><variable>ownaccount</variable><locvar>card</locvar></in>
            <out><value>transfer</value><variable>pwd</variable>
                        <variable>amount</variable><variable>ownaccount</variable><variable>account</variable>
                        <locval>settop</locval></out>
        </actions>
    </process>
    <process id="P_settop">
            <in><value>transfer</value><input>pwd</input>
                        <input>amount</input><input>ownaccount</input><input>account</input>
                        <locval>settop</locval></in>
            <in><value>password</value><variable>pwd</variable></in>
            <out><value>transfer</value><variable>pwd</variable>
                        <variable>amount</variable><variable>ownaccount</variable><variable>account</variable>
                        <locval>settop</locval></out>
    </process>
    <process id="P_server">
            <in><value>transfer</value><input>pwd</input>
                        <input>amount</input><input>ownaccount</input><input>account</input></in>
            <out><value>transfer</value><variable>account</variable>
                        <variable>pwd</variable><variable>amount</variable><locvar>ownaccount</locvar></out>
    </process>
  </processes>
```

4.1.7. Predicates

Last but not least, the model can contain predicates, which can be referred to from, e.g., policies. Currently, predicates have arity 2 and can relate an actor, who comes in second position, to another actor. For example in the setting below, Margarethe trusts Charlie, and Fred and Charlie are Employees, Margarethe is a customer, and Fritz is a technician:

```
<predicates>
      <predicate id="trusts" arity="2">
        <value>Margarethe Charlie</value>
      </predicate>
      <predicate id="role" arity="2">
        <value>Employee Fred</value>
        <value>Employee Charlie</value>
        <value>Customer Margarethe</value>
        <value>Technician Fritz</value>
      </predicate>
</predicates>
```

4.1.8. Scenario

The goal of the company in this scenario is to forbid the employee to use an ATM card. This can be modelled as forbidding an input action by an employee using a card that is owned by a customer:

```
<actionGoal attacker="Z" profit="1000">
  <credentials>
    <credPredicate name="role"><value>Employee</value><variable>Z</variable></credPredicate>
    <credItem name="Card"><credData name="Owner"><variable>O</variable></credData></credItem>
    <credPredicate name="role"><value>Customer</value><variable>O</variable></credPredicate>
  </credentials>
  <enabled><in /></enabled>
</actionGoal>
```

4.2. Telecommunication

Telecommunications products and services are functional in a complex environment involving a multitude of different interconnected networks, service providers and network operators with opposed financial interests acting in highly competitive markets offering complex products and services.

Due to the market structure described above, new customers are not easily available, but normally need to be lured away from competitors. As the providers try to escape the pricing pressure resulting from replaceability of the communication goods, e.g. new tariffs more and more become a mixture of free (flat rate) components being heavily advertised and, less noticeable, much more expensive components for compensation and revenue generation. New products need to be launched under significant time pressure resulting from strong competition in the market, leaving little time and space to account for potential misuse of the product. Apart from that, often the misuse resulting from product design flaws is a learning process which takes place once the respective product or service has been launched into the market.

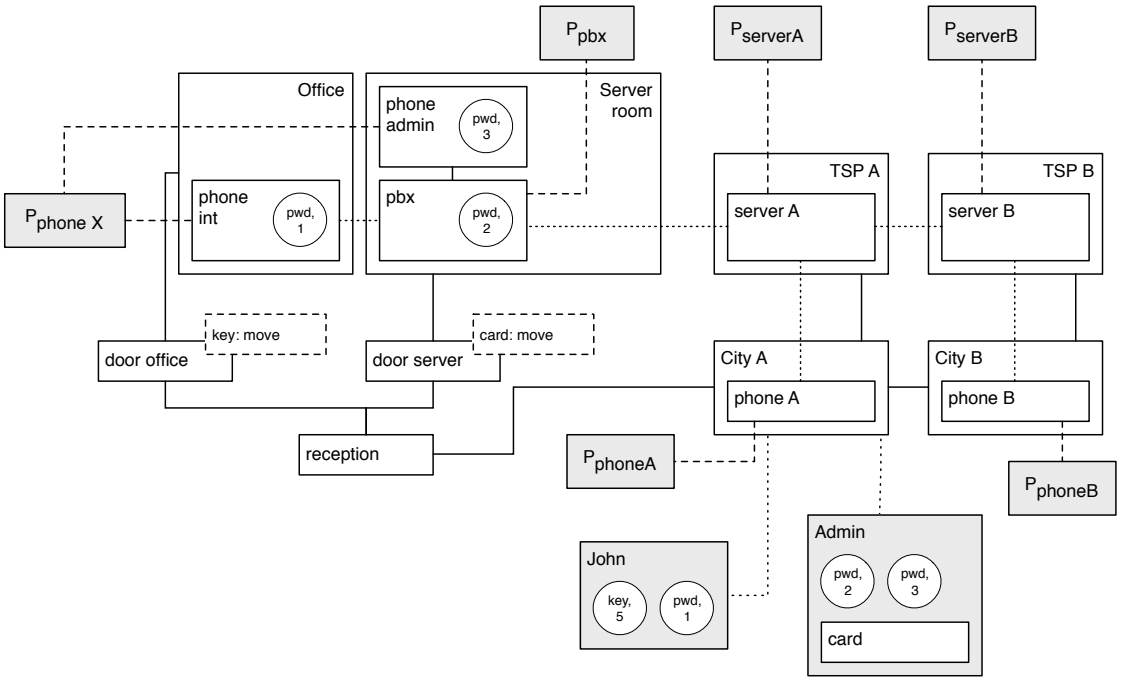

Figure 4.3.: A model for a case in the telecommunication case study. For an explanation of the meaning of the shapes please refer to Figure 4.1.

The above tendency, in contrast, encourages cherry-picking among customers of Telco companies. This is especially true for so-called knowledge insiders that know the market very well, trying to make as much use of (or monetary gain from) the products offered as possible.

An example of the sort of fraud addressed by the project involves using insider knowledge to perform calls without having to pay for them. This can for example occur by obtaining the necessary credentials at user phones, or by hacking the telephone system at a company.

4.2.1. The Model

In the model for the telecommunication case we model a company with a telephone system that connects to a telecommunication service provider TSP A, which in turn connects to a TSP B. Internal calls never leave the company, while calls in the domain of TSP A are handled by its communication system, and only external calls are transferred to TSP B.

Figure 4.3 shows that model for the telecommunication case, and Figure 4.4 shows the Attack Navigator Map for this model.

In the modelled case there exist two internal phones that are protected by a password to be provided in order to make a call, and two external phones in the domains of TSP A

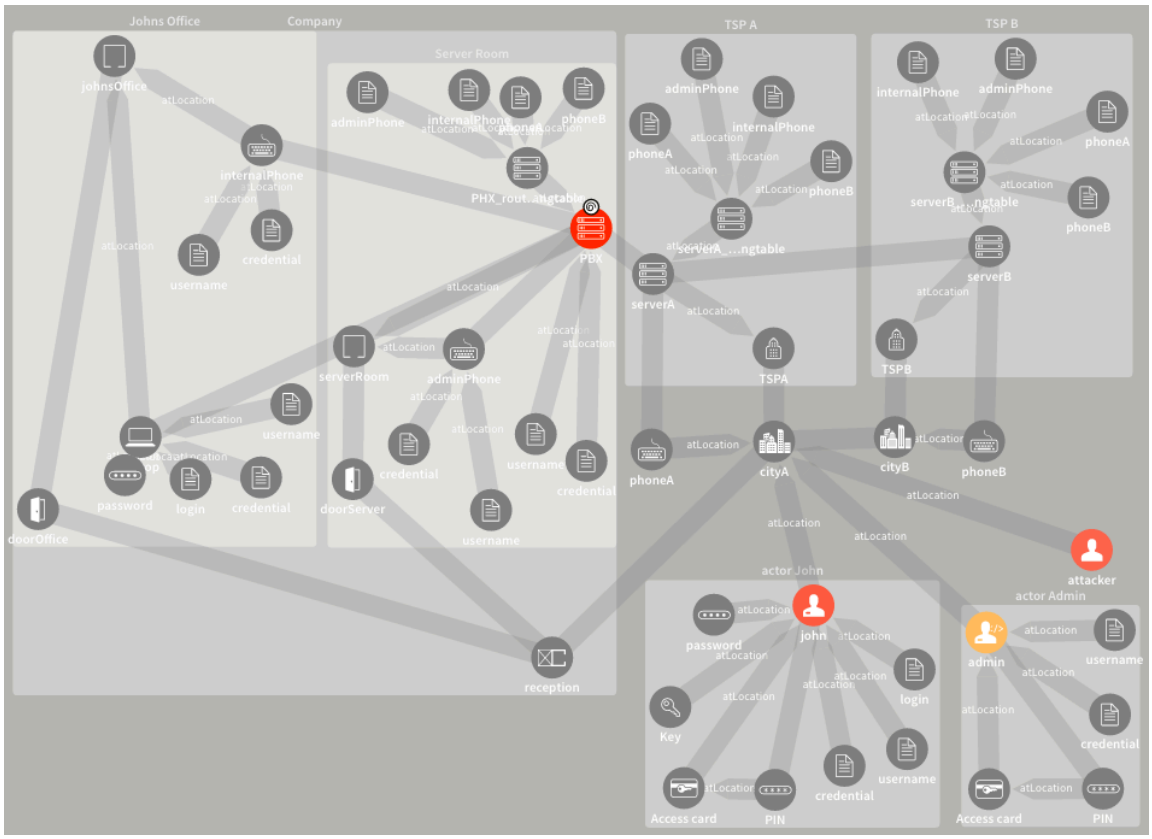

Figure 4.4.: The Attack Navigator Map for the telecommunication case study model from Figure 4.3.

and B, respectively. The doors to the office and the server room are protected by a key and a card that are required by the policies at the doors. The telecommunication infrastructure in this example is not protected by any policies, instead, the processes at the nodes will require a password, where applicable.

4.2.2. Social Layer

On the social layer, the actors in the system are John, who is an employee in the modelled organisation, Admin, and an attacker, represented in the XML file as

```
<actors>
  <actor id="john" name="john"><atLocations>world</atLocations></actor>
  <actor id="admin" name="admin"><atLocations>world</atLocations></actor>
  <actor id="attacker" name="attacker"><atLocations>world</atLocations></actor>
</actors>
```

A useful approach is to classify users by their role, access rights, and other determining properties, and then to create one actor per equivalence class of properties. The actors are all created at the node representing the outside of the organisation. Again, it is useful to apply this modeling to avoid that actors might be located at a place in the model that they do not have access to.

4.2.3. Physical Layer

On the physical layer, there exist locations relevant to the case in question; as for actors, it is a good idea to identify locations with relevant properties and only include locations from these equivalence classes.

In the model, there exist locations for John's office, the server room and a corridor in the organisation, the outside world, and two telecommunication service providers, A and B. Using the above reasoning, TSP B represents *all* TSPs other than the one that the company is connected to directly:

```
<locations>
  <location id="johnsOffice"  name="johnsOffice"/>
  <location id="serverRoom"  name="serverRoom"/>
  <location id="reception"  name="reception"/>
  <location id="doorOffice"  name="doorOffice"/>
  <location id="doorServer"  name="doorServer"/>
  <location id="world"  name="world"/>
  <location id="TSPA"  name="TSPA"/>
  <location id="TSPB"  name="TSPB"/>
</locations>
```

The model also contains a number of items and data that belong to the physical layer, for example, keys, access cards, etc:

```
<data id="password" name="password" value="john" ><atLocations>john laptop</atLocations></data>
<data id="login" name="login" value="john" ><atLocations>john laptop</atLocations></data>
<data id="credential" name="credential" value="john" ><atLocations>john internalPhone laptop</atLocations></data>
<data id="username" name="username" value="john" ><atLocations>john internalPhone laptop</atLocations></data>
<data id="credential-2" name="credential" value="admin" ><atLocations>PBX admin adminPhone</atLocations></data>
<data id="username-2" name="username" value="admin" ><atLocations>admin PBX adminPhone</atLocations></data>
<item id="key" name="key" ><atLocations>john</atLocations></item>
<item id="card" name="AccessCard" ><atLocations>john</atLocations></item>
<data id="pin" name="PIN" value="42" ><atLocations>card john</atLocations></data>
<item id="card-2" name="AccessCard" ><atLocations>admin</atLocations></item>
<data id="pin-2" name="PIN" value="43" ><atLocations>card-2 admin</atLocations></data>
```

Just like the seperation between data and items, the separation between the virtual and the physical layer is not always clear: here, we decided to list the server, laptop, and phones as elements of the virtual layer. From the viewpoint of the model, this does not make a difference, from the viewpoint of the modeller it may.

4.2.4. Virtual Layer

We assume the phone system to be Internet-based, so the phones, the pbx, and the servers at the two TSPs are network based. John's laptop is modelled in the system to illustrate the interaction between the physical and the virtual layer.

```
<item id="PBX" name="PBX" ><atLocations>serverRoom</atLocations></item>
<item id="internalPhone" name="internalPhone" ><atLocations>johnsOffice</atLocations></item>
<item id="laptop" name="laptop" ><atLocations>johnsOffice</atLocations></item>
<item id="phoneA" name="phoneA" ><atLocations>world</atLocations></item>
<item id="phoneB" name="phoneB" ><atLocations>world</atLocations></item>
<item id="adminPhone" name="adminPhone" ><atLocations>world</atLocations></item>
<item id="serverA" name="serverA" ><atLocations>TSPA</atLocations></item>
<item id="serverB" name="serverB" ><atLocations>TSPB</atLocations></item>
<item name="routingtable" id="PBX_routingtable"><atLocations>PBX</atLocations></item>
<item name="routingtable" id="serverA_routingtable"><atLocations>serverA</atLocations></item>
<item name="routingtable" id="serverB_routingtable"><atLocations>serverB</atLocations></item>
<data id="routePBX1" name="adminPhone" value="adminPhone"><atLocations>PBX_routingtable</atLocations></data>
<data id="routePBX2" name="internalPhone" value="internalPhone"><atLocations>PBX_routingtable</atLocations></data>
<data id="routePBX3" name="phoneA" value="serverA"><atLocations>PBX_routingtable</atLocations></data>
<data id="routePBX4" name="phoneB" value="serverA"><atLocations>PBX_routingtable</atLocations></data>
<data id="routeserverA1" name="internalPhone" value="PBX"><atLocations>serverA_routingtable</atLocations></data>
<data id="routeserverA2" name="adminPhone" value="PBX"><atLocations>serverA_routingtable</atLocations></data>
```

```
<data id="routeserverA3" name="phoneA" value="phoneA"><atLocations>serverA_routingtable</atLocations></data>
<data id="routeserverA4" name="phoneB" value="serverB"><atLocations>serverA_routingtable</atLocations></data>
<data id="routeserverB1" name="internalPhone" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
<data id="routeserverB2" name="adminPhone" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
<data id="routeserverB3" name="phoneA" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
<data id="routeserverB4" name="phoneB" value="phoneB"><atLocations>serverB_routingtable</atLocations></data>
```

The last items and data entries represent routing information for phone calls, or more generally IP packets. The routing tables are located at the pbx and the two servers, and the entries represent at each of these locations, where packages for all phones in the system should be routed. In the real world, this routing is based on a combination of the phone number and, for cellphones, where the phone is located. Assuming stationary phones, the routing information just contains the next hop of the route, just like IP routing tables.

4.2.5. Policies

The policies restrict access to the phones and the rooms. In order to enter John's office, one needs a key that John possesses,

```
<policy id="johnsOffice_door">
  <atLocations>doorOffice</atLocations>
  <credentials><credItem name="key"/></credentials>
  <enabled><move/></enabled>
</policy>
```

and to enter the server room one needs a card and a matching pin

```
<policy id="serverRoom_door">
  <atLocations>doorServer</atLocations>
  <credentials>
    <credItem name="AccessCard"><credData name="PIN"><variable>Y</variable></credData></credItem>
    <credData name="PIN"><variable>Y</variable></credData>
  </credentials>
  <enabled><move /></enabled>
</policy>
```

For using the phones, one needs the correct credentials and a username, all of which are modelled as data, and for the laptop also a username and a password to login

```
<policy id="phone_call">
  <atLocations>internalPhone</atLocations>
  <credentials>
    <credData name="credential"><value>john</value></credData>
    <credData name="username"><value>john</value></credData>
  </credentials>
  <enabled>
    <out><tuple><value>call</value><variable>callee</variable></tuple></out>
  </enabled>
</policy>
<policy id="laptop_call">
  <atLocations>laptop</atLocations>
  <credentials>
    <credData name="password"><value>john</value></credData>
    <credData name="login"><value>john</value></credData>
    <credData name="credential"><value>john</value></credData>
    <credData name="username"><value>john</value></credData>
  </credentials>
  <enabled>
    <out><tuple><value>call</value><variable>callee</variable></tuple></out>
  </enabled>
</policy>
<policy id="PBX_call">
  <atLocations>PBX</atLocations>
  <credentials>
    <credData name="credential"><value>admin</value></credData>
    <credData name="username"><value>admin</value></credData>
  </credentials>
  <enabled>
```

```
        <out><tuple><value>call</value><variable>callee</variable></tuple></out>
      </enabled>
    </policy>
  </policies>
```

4.2.6. Processes

In the model we show two kinds of processes: at a higher level those that realise the calling functionality, and at a lower level those that implement IP packages and traffic. We will now discuss these processes in more detail, using their abstract representation.

The process at the pbx waits for a call to be made or for an incoming call; the name of the target phone location is used as number. A call is sent as an IP packet, an incoming call is just answered to simulate a phone call:

$$\begin{aligned} P_{pbx\_call} &:= \mathbf{in}(\texttt{'call'}, !callee).\\ &\quad\ \mathbf{out}(\texttt{'IP'}, \texttt{'PBX'}, callee, (\texttt{'ring'}, \texttt{'PBX'}, callee))\\ P_{pbx\_answer} &:= \mathbf{in}(\texttt{'ring'}, \texttt{'PBX'}, !caller).\\ &\quad\ \mathbf{out}(\texttt{'IP'}, \texttt{'PBX'}, caller, (\texttt{'answer'}, \texttt{'PBX'}, caller)) \end{aligned}$$

The IP packages, including routing, are handled at the individual nodes. The route information is assumed to be stored at the pbx and the servers, and as discussed above contains information about internal and external numbers. We assume this information to be explicit for all numbers; adding prefix matching would enable modelling of real world routing. For the scenario, the routing information at the PBX consists of pairs of target node and hop to take: $(\texttt{phoneint}, \texttt{phoneint})$, $(\texttt{phoneadmin}, \texttt{phoneadmin})$, $(\texttt{phoneA}, \texttt{serverA})$, and $(\texttt{phoneB}, \texttt{serverA})$:

$$\begin{aligned} P_{pbx\_IPin} &:= \mathbf{in}(\texttt{'IP'}, !source, \texttt{'PBX'}, !tuple).\mathbf{out}(tuple)\\ P_{pbx\_IPout} &:= \mathbf{in}(\texttt{'IP'}, !source, !target, !tuple).\\ &\quad\ \mathbf{in}(target, !route)@\texttt{PBX\_routingtable}.\\ &\quad\ \mathbf{out}(\texttt{'IP'}, !source, !target, !tuple)@uroute \end{aligned}$$

The process P_{pbx_IPout} does handle both traffic originating from the PBX and traffic "through" the PBX to internal phones.

The processes at the servers A and B are slightly simpler than the one at the pbx, since no calls can be started from here; consequently we only need the process for routing IP traffic:

$$\begin{aligned} P_{serverX\_IP} &:= \mathbf{in}(\texttt{'IP'}, !source, !target, !tuple).\\ &\quad\ \mathbf{in}(target, !route)@\texttt{serverX\_routingtable}.\\ &\quad\ \mathbf{out}(\texttt{'IP'}, !source, !target, !tuple)@uroute \end{aligned}$$

At the servers, we have routing information such that server B sends all calls but the ones for phoneB to server A, and server A sends calls for phone B to server B, for phone A to phone A, and for the internal phones to the pbx. As discussed, this is encoded in the routing information at the servers.

At the phones, also the same processes as at the PBX are executed, with the exception of the process for outgoing IP traffic; since phones do not act as routers, the *IPout* process is simpler:

$$P_{phoneX\_IPout} := \mathbf{in}(\texttt{'IP'}, \texttt{'phoneX'}, !target, !tuple).\ \mathbf{out}(\texttt{'IP'}, \texttt{'phoneX'}, !target, !tuple)@\texttt{router}$$

The location *router* is the next hop, which is either the *PBX*, *serverA*, or *serverB*.

This model can be extended to also include the accounting information; whenever a call reaches serverA, the server process could, based on the target location of the call, output a call termination fee to the called TSP, charge the caller, or check that a contract is being paid for.

4.2.7. Scenario

To model the attack described above, the goal of an attacker would be to output the tuple $(\texttt{'ring'}, \texttt{X}, \texttt{phoneB})$ at `phoneB`, where X would be a phone inside the organisation:

```
<locationGoal attacker="actor__attacker" profit="5000">
  <asset>
    <tuple><value>ring</value><value>internalPhone</value><value>phoneB</value></tuple>
    <tuple><value>ring</value><value>laptop</value><value>phoneB</value></tuple>
    <tuple><value>ring</value><value>PBX</value><value>phoneB</value></tuple>
  </asset>
  <location>phoneB</location>
</locationGoal>
```

4.3. Cloud

Cloud computing has gained remarkable popularity in recent years due to the economic and technical advantages of this new way of delivering computing resources. Customers benefit from rapid provisioning and seemingly infinite scalability, while only being charged on a pay-per-use basis. Computing resources can be provided on different abstraction layers (Mell & Grance, 2009b), where the lowest one provides basic resources (servers, network, and storage), and higher ones provide applications to the end-users (e.g., Google's GMail, Salesforce.com). In this case-study we are focusing on the lowest abstraction layer, that is *Infrastructure-as-a-Service* or *Infrastructure Clouds*, since it is the most generic layer and higher ones often build upon this layer.

Although the benefits of cloud computing are evident and users demand cloud services, security is a major inhibitor (Mell & Grance, 2009a). An analysis of risks and threats in cloud computing has been conducted in (Cloud Security Alliance, 2010) and (ENISA, 2009). In particular, both reports agree that insider attacks and malicious insiders are a major risk and are among the top 10 threats. The risk is amplified due to the disappearance of physical boundaries that makes it very challenging to define a security perimeter that divides insiders from outsiders (Hay, Nance, & Bishop, 2011; Pieters, 2011b).

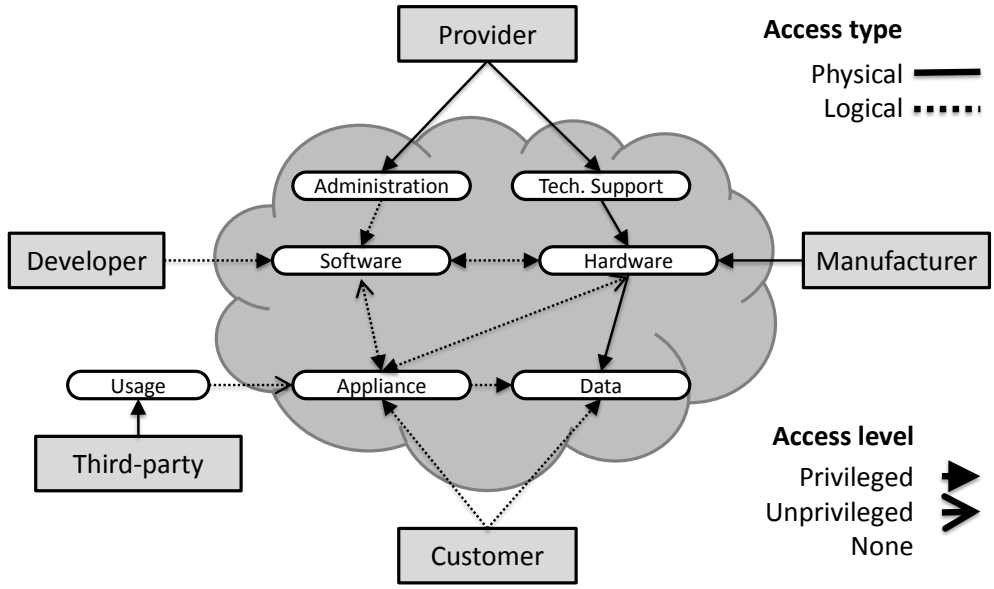

Figure 4.5.: Overview of entities and components in an infrastructure cloud model.

Figure 4.5 provides an overview of the entities and components involved in a model of an infrastructure cloud. As compared to a typical IT department within an organisation, the provider of the cloud services in such a shared infrastructure cloud is a new additional and powerful party. The multi-tenancy (different customers, including competing organisations, using the same cloud services) lead also to an unprecedented sharing of computing and infrastructure resources. The cloud provider therefore becomes an important additional factor (and risk), as it has full physical and logical access to all resources across the different consumers. In addition, the infrastructure itself is dynamic and flexible in response to the requirements of the customers.

If risk assessment in complex technical infrastructures is already difficult, it is even more complicated when human factors and physical infrastructure are added to the setup, which are correspondingly often ignored (Probst & Hunker, 2009).

The special interest to investigate this scenario within the TRE$_S$PASS project is therefore to develop models and processes that support risk assessment in complex organisations *including* human factors and physical infrastructure. The goal of this support is to simplify the identification of possible attacks and to provide qualified assessment and ranking of attacks based on factors such as the expected impact.

For cloud infrastructures, the TRE$_S$PASS model distinguishes components at a level of abstraction that corresponds well to security-relevant control points in these domains, enabling the discovery and analysis of potential attacks that exploit their connectivity. Using the model, one can formalise typical components in cloud infrastructures and their interrelationships. These include network components like switches, routers, firewalls; virtual and physical servers; actors, including administrators, users, and attackers; location details that represent rooms, doors, and other physical consideration. Because these com-

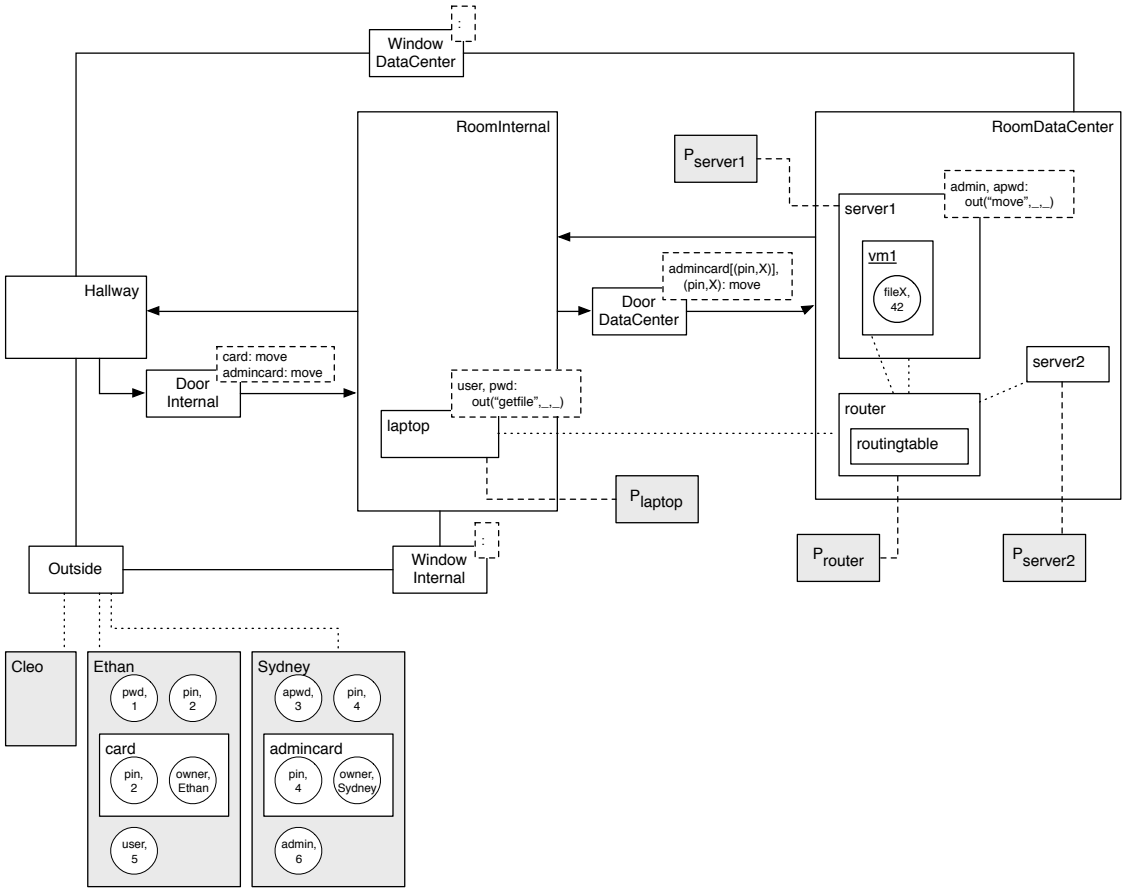

Figure 4.6.: A model for the cloud case study. For an explanation of the meaning of the shapes please refer to Figure 4.1.

ponent models show how actions on one element influence other elements, they can be combined with the connectivity relations to form an implicit search-space of all possible activity paths in the system.

4.3.1. The Model

Figure 4.6 shows a model for one scenario from the cloud case study. There are three actors with different credentials: Grey, an outsider without any credentials, Ethan, an office worker, with credentials to enter the internal room and for the laptop, and Finn, an administrator with credentials to enter the server room and for the server.

The two windows in the model illustrate the use of policies: the policies at the windows forbid all actions; however, for the attack generation they are a way of entering the internal room from the outside, and the server room from the inside.

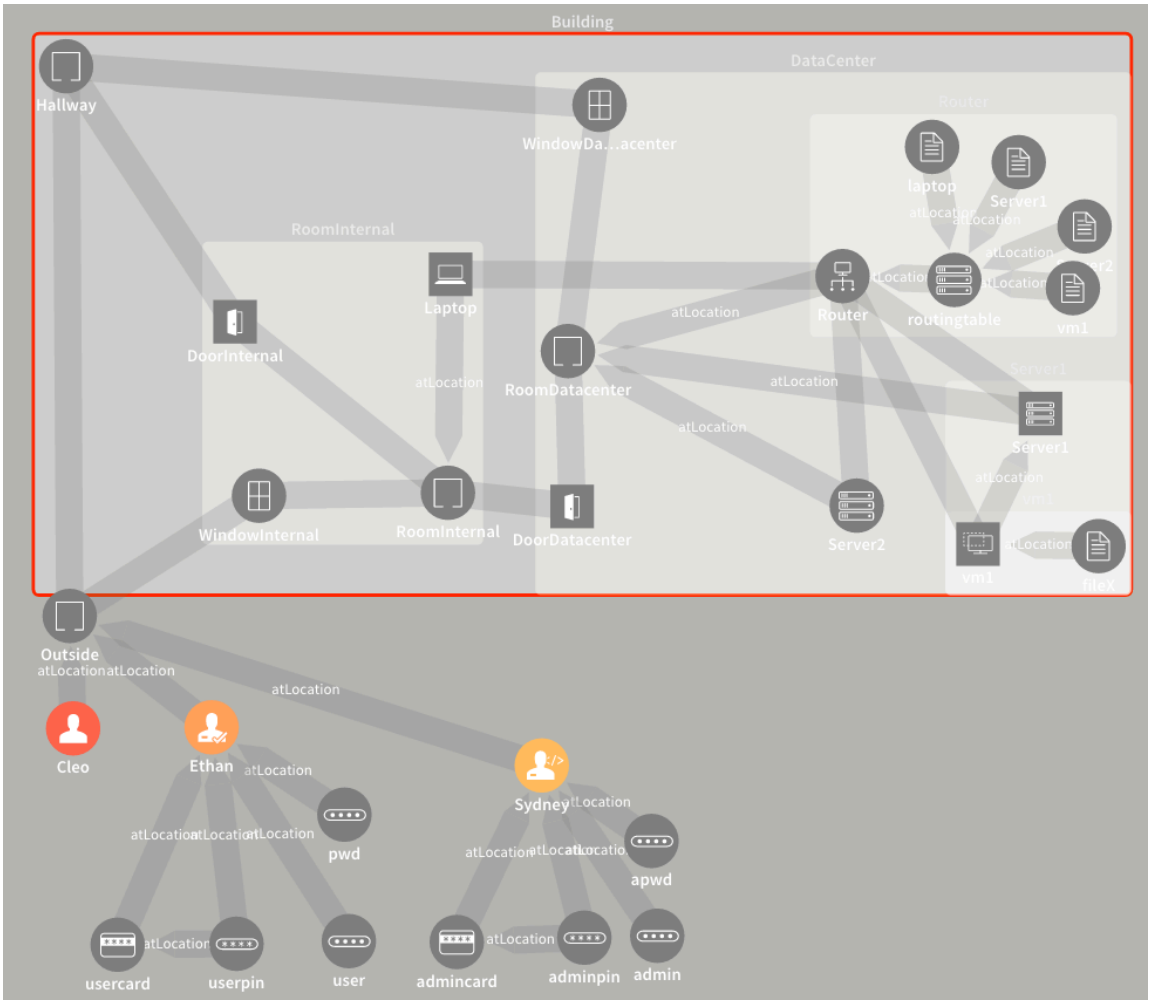

Figure 4.7.: The Attack Navigator Map for the cloud model from Figure 4.6.

The cloud case study also illustrates containment of assets: the file `fileX` is contained in the virtual machine item `vm1`, which is contained in the item `server1`. In the model these are represented with the *locatedAt* attribute.

4.3.2. Social Layer

On the social layer we model actors with different roles as described before: Cleo, a user with no access rights, Ethan, an employee with access to the internal room and to the laptop, and Sydney, a system administrator with access to the datacenter and the servers:

```
<actors>
  <actor id="Ethan"><atLocations>Outside</atLocations></actor>
  <actor id="Sydney"><atLocations>Outside</atLocations></actor>
  <actor id="Cleo"><atLocations>Outside</atLocations></actor>
</actors>
```

4.3.3. Physical Layer

At the physical layer, the main locations are the rooms, doors, and windows:

```
<location id="Outside"/>
<location id="Hallway"/>
<location id="DoorInternal"/>
<location id="RoomInternal"/>
<location id="WindowInternal"/>
<location id="DoorDatacenter"/>
<location id="RoomDatacenter"/>
<location id="WindowDatacenter"/>
```

In this scenario, the locations modelling the outside and the hallway are connected by an undirected edge, and as we will see later they do not have any policies or processes located at them:

```
<edges>
  <edge directed="false"><source>Outside</source><target>Hallway</target></edge>
  <edge><source>Hallway</source><target>DoorInternal</target></edge>
  <edge><source>DoorInternal</source><target>RoomInternal</target></edge>
  <edge><source>RoomInternal</source><target>Hallway</target></edge>
  <edge directed="false"><source>RoomInternal</source><target>WindowInternal</target></edge>
  <edge directed="false"><source>Outside</source><target>WindowInternal</target></edge>
  <edge directed="false"><source>RoomInternal</source><target>DoorDatacenter</target></edge>
  <edge directed="false"><source>DoorDatacenter</source><target>RoomDatacenter</target></edge>
  <edge directed="false"><source>WindowDatacenter</source><target>RoomDatacenter</target></edge>
  <edge directed="false"><source>Hallway</source><target>WindowDatacenter</target></edge>
  <edge directed="false"><source>Laptop</source><target>Router</target></edge>
  <edge directed="false"><source>Router</source><target>Server1</target></edge>
  <edge directed="false"><source>Router</source><target>Server2</target></edge>
  <edge directed="false"><source>Router</source><target>vm1</target></edge>
</edges>
```

In cases like this, the modeller might consider to merge these two locations to reduce the model size.

Finally, the model defines the cards for the user and the system admin to enter the doors:

```
<item name="usercard" id="usercard"><atLocations>Ethan</atLocations></item>
<data name="userpin" id="userpin" value="1"><atLocations>Ethan usercard</atLocations></data>
<data name="username" id="user" value="user"><atLocations>Ethan</atLocations></data>
<data name="password" id="pwd" value="pwd"><atLocations>Ethan</atLocations></data>
<item name="admincard" id="admincard"><atLocations>Sydney</atLocations></item>
<data name="adminpin" id="adminpin" value="1"><atLocations>Sydney admincard</atLocations></data>
<data name="username" id="admin" value="admin"><atLocations>Sydney</atLocations></data>
<data name="password" id="apwd" value="apwd"><atLocations>Sydney</atLocations></data>
```

As in the IPTV case, the pins are located both at the card and the user; the policy at the door will require the actor to have both the card and a matching pin. The matching is enforced via the unique identifier.

4.3.4. Virtual Layer

At the virtual layer, the model contains the computer equipment, network equipment, and the virtual machine:

```
<item name="Laptop" id="Laptop"><atLocations>RoomInternal</atLocations></item>
<item name="Router" id="Router"><atLocations>RoomDatacenter</atLocations></item>
<item name="routingtable" id="routingtable"><atLocations>Router</atLocations></item>
<data id="route1" name="Laptop" value="Laptop"><atLocations>routingtable</atLocations></data>
<data id="route2" name="Server1" value="Server1"><atLocations>routingtable</atLocations></data>
<data id="route3" name="Server2" value="Server2"><atLocations>routingtable</atLocations></data>
<data id="route4" name="vm1" value="vm1"><atLocations>routingtable</atLocations></data>
<item name="Server1" id="Server1"><atLocations>RoomDatacenter</atLocations></item>
<item name="Server2" id="Server2"><atLocations>RoomDatacenter</atLocations></item>
<item name="vm1" id="vm1"><atLocations>Server1</atLocations></item>
<data name="fileX" id="fileX" value="42"><atLocations>vm1</atLocations></data>
```

4.3.5. Policies

The main use of policies is the access control to the rooms by requiring either a user card or an admin card, and the control, who is allowed to start processes at the laptop and the server.

At the door to the internal room, this policy models that an actor needs either a user card or an admin card and a matching pin:

```
<policy id="p001">
  <credentials>
    <credItem name="usercard"><credData name="userpin"><variable>Y</variable></credData></credItem>
    <credData name="userpin"><variable>Y</variable></credData>
  </credentials>
  <enabled><move /></enabled>
  <atLocations>DoorInternal</atLocations>
</policy>
<policy id="p002">
  <credentials>
    <credItem name="admincard"><credData name="adminpin"><variable>Y</variable></credData></credItem>
    <credData name="adminpin"><variable>Y</variable></credData>
  </credentials>
  <enabled><move /></enabled>
  <atLocations>DoorInternal DoorDatacenter</atLocations>
</policy>
```

At the laptop, the user is allowed to start an ftp client by outputting a “getfile” request:

```
<policy id="p003">
  <credentials>
    <credData name="password"><value>pwd</value></credData>
    <credData name="username"><value>user</value></credData>
  </credentials>
  <enabled><out><tuple><value>getfile</value><wildcard/><wildcard/></tuple></out></enabled>
  <atLocations>Laptop</atLocations>
</policy>
```

This policy requires the actor to have (or know) a password with value “pwd” and a username with value “user”.

Similarly, at the server the administrator is allowed to start the process for virtual machine migration:

```
<policy id="p004">
  <credentials>
    <credData name="password"><value>apwd</value></credData>
    <credData name="username"><value>admin</value></credData>
  </credentials>
  <enabled><out><tuple><value>move</value><wildcard/><wildcard/></tuple></out></enabled>
  <atLocations>Server1 Server2</atLocations>
</policy>
```

The policies at the window are special, since they do not require any credentials, nor do they enable any actions:

```
<policy id="p005">
  <credentials></credentials>
  <enabled></enabled>
  <atLocations>WindowInternal WindowDatacenter</atLocations>
</policy>
```

This policy models that actors can not move through the window; the attack generation will nevertheless generate attacks that include this move by forcing the window open.

4.3.6. Processes

A significant part of the processes deals with IP traffic as described in the previous section. Instead of repeating this discussion here, we concentrate on the two cloud-specific processes, the ftp client and server and the migration of virtual machines.

4.3.6.1. An FTP server

The ftp server runs on `vm1` and can be accessed from the laptop. At the laptop, the process waits for a `'getfile'` request:

$$\begin{aligned} P_{laptop\_Query} \quad := \quad & \mathbf{in}(\texttt{'getfile'}, !server, !filename). \\ & \mathbf{out}(\texttt{'IP'}, \texttt{laptop}, server, (\texttt{'get'}, \texttt{laptop}, filename)). \\ & \mathbf{in}((\texttt{'file'}, filename, !content).\mathbf{out}(filename, content) \end{aligned}$$

When a `'getfile'` request is received, the user also outputs the server name, from which the file should be fetched, and the filename. This request is wrapped into an IP message and output. As discussed before, IP messages mimic the structure of IP packages: they have a header `'IP'`, a sender location and a recipient location, and a tuple that represents the message. In this process, the sender is the `laptop` and the recipient is the server specified by the user. The message is a tuple $(\texttt{'get'}, \texttt{laptop}, filename)$, specifying the command, the location where the file should be sent, and the name of the filename provided by the user.

Once the IP traffic has been routed to the virtual machine at `vm1`, a process waits for a request and serves the file:

$$\begin{aligned} P_{vm1\_FTP} \quad := \quad & \mathbf{in}((\texttt{'get'}, !sender, !filename)).\mathbf{in}(filename, !content). \\ & \mathbf{out}(\texttt{'IP'}, \texttt{vm1}, sender, (\texttt{'file'}, filename, content)) \end{aligned}$$

The process inputs the content of the file and sends a reply to the sender as IP package with the message $(\texttt{'IP'}, filename, content$. The process at the laptop waits for this message and outputs the file at the laptop.

In the model file, these processes are represented as

```
<process id="P_query">
  <actions>
    <in><tuple><value>getfile</value><input>Server</input><input>filename</input></tuple></in>
    <out><value>IP</value><value>Laptop</value><variable>Server</variable>
          <tuple><value>get</value><value>Laptop</value><variable>filename</variable></tuple></out>
    <in><tuple><value>file</value><variable>filename</variable><input>content</input></tuple></in>
    <out><tuple><variable>filename</variable><variable>content</variable></tuple></out>
  </actions>
  <atLocations>Laptop</atLocations>
</process>
<process id="P_vm1_ftp">
  <actions>
    <in><tuple><value>get</value><input>sender</input><input>filename</input></tuple></in>
    <in><variable>filename</variable><input>value</input></in>
    <out><value>IP</value><value>vm1</value><variable>sender</variable>
      <tuple><value>file</value><variable>filename</variable><variable>value</variable></tuple></out>
  </actions>
<atLocations>vm1</atLocations>
</process>
```

4.3.6.2. Virtual Machine Migration

Another process of interest for the cloud case study is the migration of virtual machines. In order to start this process, the administrator outputs a tuple at the server with the command `'move'`, the name of the virtual machine, and the target server. As a results, the process inputs the virtual machine and sends it as an IP packet to the server, where it is deserialized.

$$\begin{aligned} P_{server1Move} \quad := \quad & \mathbf{in}(\texttt{'move'}, !server, !vmname).\mathbf{in}(vmname, content). \\ & \mathbf{out}(\texttt{'IP'}, \texttt{laptop}, server, (\texttt{'vm'}, vmname, content)) \end{aligned}$$

The resulting IP packet is then routed to the recipient server by the processes described above, and at the receiving server it is re-instantiated:

$$P_{server2VM} \quad := \quad \mathbf{in}(\texttt{'vm'}, !vmname, !content).\mathbf{out}(vmname, content)$$

In the model file, these processes are represented as

```
<process id="P_server1_move">
  <actions>
    <in><tuple><value>move</value><input>server</input><input>vmname</input></tuple></in>
    <in><variable>vmname</variable><input>content</input></in>
    <out><value>IP</value><value>Server1</value><variable>server</variable>
        <tuple><value>vm</value><variable>vmname</variable><variable>content</variable></tuple></out>
  </actions>
  <atLocations>Server1</atLocations>
</process>
<process id="P_server1_receive">
  <actions>
    <in><tuple><value>vm</value><input>vmname</input><input>content</input></tuple></in>
    <out><variable>vmname</variable><variable>content</variable></out>
  </actions>
  <atLocations>Server1</atLocations>
</process>
```

4.3.7. Scenario

In this scenario, the file of interest is *fileX*, so we use an asset goal:

```
<assetGoal attacker="X">
  <asset>fileX</asset>
</assetGoal>
```

4.4. ATM

ATM machines are composed of a money safe and a computer that controls the ATM's devices (screen, keyboard, printer, network interfaces, money safe mechanisms, etc.) through software programs. Most ATM computers are composed of outdated hardware running legacy operating systems, e.g. Windows NT, which are not supported by the vendors or by the anti-virus providers. The installation of the ATM machines varies as there are well protected ATMs installed inside bank branches, while others are deployed in the street and some are not even embedded in a wall.

ATM attacks are common and include classic physical attacks and emerging digital attacks. Examples include:

- Physical attacks where the attacker physically steels the ATM to open the safe and take the money.
- Digital attacks where the attacker installs a malware agent into the operating system to take control of the devices including the ability to withdraw money from the safe through the device' s interfaces.

In order to perform a proper risk assessment of the ATM network as a whole, four types of data must be considered and combined into a single unified model, suitable to be processed on a consolidated data schema:

- Technical data (about the machines)
- Environmental data (about the territory)
- Historical data (about past occurrences)

Attacks do not happen randomly in time and are neither equally distributed over the territory. Machines installed on certain places are more likely to be attacked than others, thus spatio-temporal hotspots exist. This case study therefore adds both the temporal and a geographic dimension to the project.

The ATM case study builds an analysis process at the macro-level based on data from the ATM network and surrounding area, and outputs the vulnerability level of each ATM. More specifically, this case study illustrates how to perform an analysis at a macro level for identifying priority areas in need of detailed analysis using the TREsPASS model and tools.

The ATM case study is about a gas station (petrol station) located in a Lisbon neighborhood (metropolitan district). The Gas station is open from 6 am to 12 pm and provides several services including: gas, car-wash, food, cash etc. Besides, the gas station has several controls in place including surveillance system, fire and burglar alarm.

Three main zones can be identified at the gas station: store zone, where the general public can transit to buy or request services including an ATM (the internet connection is isolated from the gas station internet connection and belongs to the Interbank Network provider); internal office where the technological components of the gas station are located: workstation, printer, router and local internet connection used to share information with the security provider, and toilets.

4.4.1. The Model

Figure 4.8 shows a model for a scenario in the ATM case. The attack navigator map for the general case is shown in Figure 4.9, and shows significantly more details, that mostly influence the quantitative data for actors and actions.

As both figures illustrate, a very interesting aspect of the ATM case study model is the modelling of the ATM itself. In the IPTV model presented in Section 4.1 the ATM was modelled only as a location with a safe, where the safe had a policy that required a card

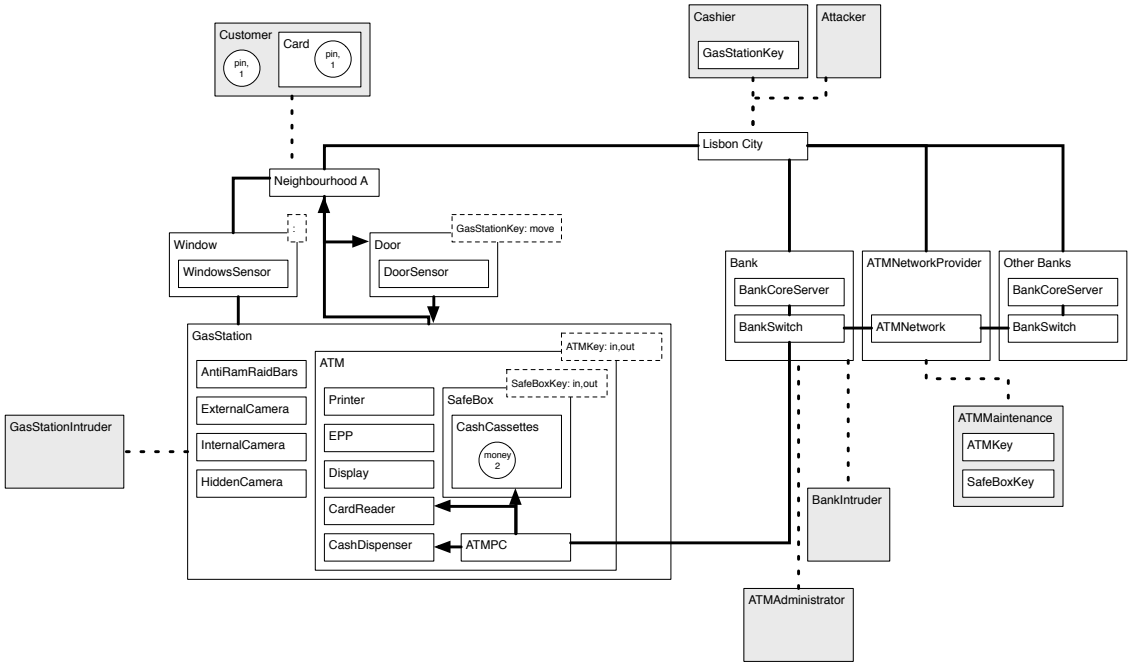

Figure 4.8.: A model for the ATM case study. For an explanation of the meaning of the shapes please refer to Figure 4.1.

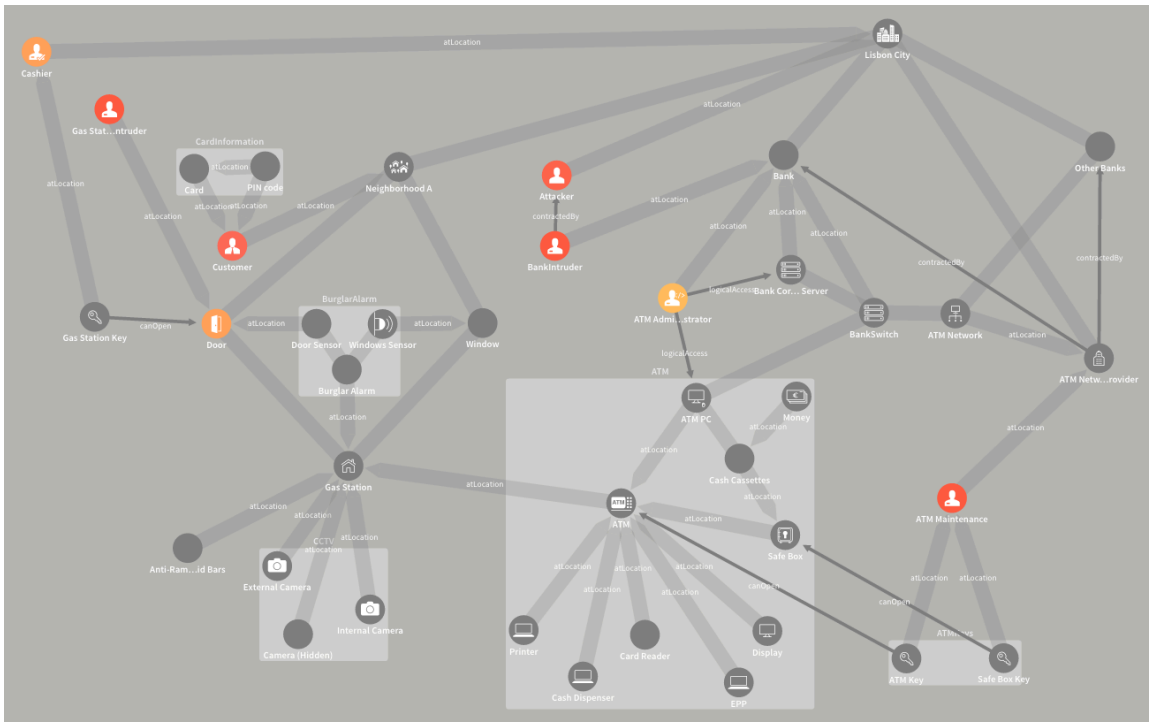

Figure 4.9.: The Attack Navigator Map for the ATM model from Figure 4.8.

with a matching pin in order to input money: $card[(pin, X)], (pin, X) : \mathbf{in}$. This approach will be significantly refined here.

4.4.2. Social Layer

Most actor in the model share the same level of access and information – none. They are present in the model mostly to indicate, based on their location, that they have privileged access, for example, because they already have infiltrated an organisation or a building. The only actors with special knowledge are the customer, who has an ATM card and a matching pin, the ATM maintenance staff, who has keys for the ATM and the safe box, and the cashier, who has a key for the gas station:

```
<actor id="Cashier" name="Cashier"><atLocations>LisbonCity</atLocations></actor>
<actor id="GasStationIntruder" name="GasStationIntruder"><atLocations>GasStation</atLocations></actor>
<actor id="Customer" name="Customer"><atLocations>NeighborhoodA</atLocations></actor>
<actor id="ATMMaintenance" name="ATMMaintenance"><atLocations>ATMNetworkProvider</atLocations></actor>
<actor id="ATMAdministrator" name="ATMAdministrator"><atLocations>Bank</atLocations></actor>
<actor id="Attacker" name="Attacker"><atLocations>LisbonCity</atLocations></actor>
<actor id="BankIntruder" name="BankIntruder"><atLocations>Bank</atLocations></actor>
```

4.4.3. Physical Layer

On the physical layer, the ATM model represents the banks, the ATM, and the gas station, as well as, for example, the city and neighbourhood, where the ATM is located:

```
<locations>
  <location id="GasStation" name="GasStation"/>
  <location id="NeighborhoodA" name="NeighborhoodA"/>
  <location id="LisbonCity" name="LisbonCity"/>
  <location id="Window" name="Window"/>
  <location id="Door" name="Door"/>
  <location id="Bank" name="Bank"/>
  <location id="OtherBanks" name="OtherBanks"/>
  <location id="ATMNetworkProvider" name="ATMNetworkProvider"/>
</locations>
<assets>
  <item id="card" name="card"><atLocations>Customer</atLocations></item>
  <data id="pin" name="pin" value="42"><atLocations>card Customer</atLocations></data>
  <item id="ExternalCamera" name="ExternalCamera"><atLocations>GasStation</atLocations></item>
  <item id="InternalCamera" name="InternalCamera"><atLocations>GasStation</atLocations></item>
  <item id="HiddenCamera" name="HiddenCamera"><atLocations>GasStation</atLocations></item>
  <item id="DoorSensor" name="DoorSensor"><atLocations>Door</atLocations></item>
  <item id="AntiRamRaidBars" name="AntiRamRaidBars"><atLocations>GasStation</atLocations></item>
  <item id="ATM" name="ATM"><atLocations>GasStation</atLocations></item>
  <item id="SafeBox" name="SafeBox"><atLocations>ATM</atLocations></item>
  <item id="CashCassettes" name="CashCassettes"><atLocations>SafeBox</atLocations></item>
  <item id="Money" name="Money"><atLocations>CashCassettes</atLocations></item>
  <item id="Display" name="Display"><atLocations>ATM</atLocations></item>
  <item id="Printer" name="Printer"><atLocations>ATM</atLocations></item>
  <item id="EPP" name="EPP"><atLocations>ATM</atLocations></item>
  <item id="CardReader" name="CardReader"><atLocations>ATM</atLocations></item>
  <item id="CashDispenser" name="CashDispenser"><atLocations>ATM</atLocations></item>
  <item id="ATMKey" name="ATMKey"><atLocations>ATMMaintenance</atLocations></item>
  <item id="SafeBoxKey" name="SafeBoxKey"><atLocations>ATMMaintenance</atLocations></item>
  <item id="WindowsSensor" name="WindowsSensor"><atLocations>Window</atLocations></item>
  <item id="GasStationKey" name="GasStationKey"><atLocations>Cashier</atLocations></item>
  <item id="BurglarAlarm" name="BurglarAlarm"><atLocations>GasStation</atLocations></item>
```

4.4.4. Virtual Layer

On the virtual layer, we consider the processes, to be discussed below, and the hardware representing the computer network:

```
    <item id="BankSwitch" name="BankSwitch"><atLocations>Bank</atLocations></item>
    <item id="ATMNetwork" name="ATMNetwork"><atLocations>ATMNetworkProvider</atLocations></item>
    <item id="BankCoreServer" name="BankCoreServer"><atLocations>Bank</atLocations></item>
    <item id="ATMPC" name="ATMPC"><atLocations>ATM</atLocations></item>
  </assets>
```

4.4.5. Policies

The policies in the ATM model refer mostly to entering the gas station and accessing the ATM. For entering the gas station, the actor needs the key:

```
  <policy id="pol001">
    <credentials><credItem name="GasStationKey"/></credentials>
    <enabled><move/></enabled>
    <atLocations>Door</atLocations>
  </policy>
```

For accessing the cash cassettes, the actor needs the key for the ATM and the key for the safe box:

```
  <policy id="pol002">
    <credentials><credItem name="SafeBoxKey"/></credentials>
    <enabled><in/></enabled>
    <atLocations>safe-box</atLocations>
  </policy>
  <policy id="pol003">
    <credentials><credItem name="ATMKey"/></credentials>
    <enabled><in/></enabled>
    <atLocations>ATM</atLocations>
  </policy>
```

In comparison with the IPTV case, the most notable difference is the absence of a policy requiring a card and a matching pin. These are now checked in the process executing at the ATM, as will be discussed in the next section.

4.4.6. Processes

The most interesting aspect of the ATM case study model is the modelling of the ATM itself. In the IPTV model presented in Section 4.1, the ATM was modelled only as a location with a safe, where the safe had a policy that required a card with a matching pin in order to input money: $\mathit{card}[(\mathit{pin}, X)], (\mathit{pin}, X) : \mathbf{in}$.

Of course, this is a very coarse abstraction from the functioning of an ATM, and it does not support detection of (unfortunately) common attacks on ATMs such as holding the money back in the cash slot, or the card in the card reader, or obtaining the pin by reading it with some means from the keyboard.

A more realistic model of an ATM divides the ATM in several locations: the card reader, the keyboard, the cash slot, the money box, a computer controlling the ATM, and the ATM itself, containing all these components.

The process at the ATM first inputs the card and the pin, then it checks the pin against the card, inputs the amount to withdraw from the keyboard and the account from the card,

sends a request to the bank computer for authorization, and when it receives this, it inputs money from the money box and outputs it in the cash slot:

$$
\begin{aligned}
P_{AT} \quad := \quad & \mathbf{in}(\texttt{'card'}, !card)@\texttt{cardreader}. \\
& \mathbf{in}(\texttt{'pin'}, !userpin)@\texttt{keyboard}.\mathbf{in}(\texttt{'pin'}, userpin)@\texttt{card}. \\
& \mathbf{in}(\texttt{'account'}, !account)@\texttt{card}.\mathbf{in}(\texttt{'amount'}, amount)@\texttt{keyboard}. \\
& \mathbf{out}(\texttt{'IP'}, \texttt{ATM}, \texttt{bankcomputer}, (\texttt{'withdraw'}, account, amount)). \\
& \mathbf{in}((\texttt{'withdraw'}, \texttt{'OK'})).\mathbf{in}(\texttt{'money'}, !cash)@\texttt{moneybox}. \\
& \mathbf{out}(\texttt{'card'}, card)@\texttt{cardreader}.\mathbf{out}(\texttt{'money'}, cash)@\texttt{cashslot}
\end{aligned}
$$

This example also illustrates the coordination between process through the waiting for input: after sending the authorization request to the bank computer, the process waits for the ok and only is able to continue execution once it has received this.

Also this process, though considerably closer to reality than the model used in the IPTV case, uses abstraction, for example when checking the pin on the card. In reality, this is performed by a process being executed on the card and by communication with the bank core server.

This communication with the bank core server is again implemented via IP traffic, as described before. The process above is represented in the model as

```
<process id="process__ATM">
  <atLocations>ATM</atLocations>
  <actions>
    <in><value>card</value><input>card</input><locval>CardReader</locval></in>
    <in><value>pin</value><input>userpin</input><locval>EPP</locval></in>
    <in><value>pin</value><variable>userpin</variable><locvar>card</locvar></in>
    <in><value>account</value><input>account</input><locvar>card</locvar></in>
    <in><tuple><value>withdraw</value><input>amount</input></tuple><locval>EPP</locval></in>
    <out><value>IP</value><value>ATM</value><value>BankCoreServer</value>
        <tuple><value>withdraw</value><variable>account</variable><variable>amount</variable><value>ATM</value></tuple></out>
    <in><tuple><value>withdraw</value><value>ok</value></tuple></in>
    <in><value>Money</value><input>cash</input><locval>CashCassettes</locval></in>
    <out><variable>card</variable><locval>CardReader</locval></out>
    <out><tuple><value>Money</value><variable>cash</variable></tuple><locval>CashDispenser</locval></out>
  </actions>
</process>
```

4.4.7. Scenario

The obvious goal for an attacker in the ATM scenarion is to obtain money, so we choose an asset goal:

```
<assetGoal attacker="actor__attacker" profit="5000">
  <asset>Money</asset>
</assetGoal>
```

5. Conclusions

This deliverable presents the final result of WP1, the TRE$_S$PASS socio-technical security model and some applications to the case studies. The case studies are sufficiently different to illustrate different aspects of the modelling, especially with respect to the physical, virtual, and social layer, policies, and processes.

From a structured literature review of modelling approaches for socio-technical systems we have identified important properties of such systems; this has driven the design and development of the final model. Investigating other models has enabled us to understand limitations of the TRE$_S$PASS model, and to adapt accordingly. This has lead to a toolbox of mechanisms for extending the model and the TRE$_S$PASS tools (The TRE$_S$PASS Project, D1.3.2, 2015).

As the models in Chapter 4 show there is not a single, correct model for a given scenario. To the contrary, modelling of socio-technical systems, just like modelling in general, requires an iterative approach that in the case of TRE$_S$PASS is guided by the amount and the quality of the attacks identified. As shown for ATMs and network communication, processes and interactions, but also infrastructure, can be modelled at quite different levels of detail. The less detailed the model, the less detailed the identified attacks will be, but this can serve a purpose in getting a first understanding of the threat scenario faced.

The final TRE$_S$PASS socio-technical security model has been validated against the case studies. An important property introduced in the model and the TRE$_S$PASS tools is the separation of data storage and the model itself. Through the unique identifiers of model components, every element in the model can addressed, associated with quantitative data, but also be the target of mapping analysis results back or relating analysis results and model components for example in visualisations.

References

Al Sabbagh, B., & Kowalski, S. (2012, June). ST(CS)2 - Featuring socio-technical cyber security warning systems. In *Proceedings title: 2012 international conference on cyber security, cyber warfare and digital forensic (cybersec)* (pp. 312–316). IEEE. doi: 10.1109/CyberSec.2012.6246110

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theor. Comput. Sci.*, *126*(2), 183-235.

Bæk, C. T., & Bach, C. (2016). *Log-based identification of potential insider attackers* (Tech. Rep.). Technical University of Denmark.

Behrmann, G., David, A., & Larsen, K. G. (2004). A tutorial on UPPAALl. In *Revised lectures of the international school on formal methods for the design of computer, communication and software systems (sfm-rt 2004)* (Vol. 3185, pp. 200–236). Springer.

Cloud Security Alliance. (2010). *Top threats to cloud computing v1.0.* https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

David, A., Larsen, K. G., Legay, A., Mikucionis, M., & Wang, Z. (2011). Time for statistical model checking of real-time systems. In *Proc. of computer aided verification (cav 2011)* (Vol. 6806, pp. 349–355). Springer Verlag.

David, N. (2014). *TREsPASS project* (Unpublished master's thesis). Ecole Centrale de Nantes, IRCCYN.

David, N., David, A., Hansen, R. R., Larsen, K. G., Legay, A., Olesen, M. C., & Probst, C. W. (2015). Modelling social-technical attacks with timed automata. In E. Bertino & I. You (Eds.), *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats, MIST 2015, Denver, Colorado, USA, October 16, 2015* (pp. 21–28). ACM. Retrieved from http://doi.acm.org/10.1145/2808783.2808787 doi: 10.1145/2808783.2808787

De Nicola, R., Ferrari, G., & Pugliese, R. (1998). Klaim: a kernel language for agents interaction and mobility. *Software Engineering, IEEE Transactions on*, *24*(5), 315–330. doi: 10.1109/32.685256

De Nicola, R., Gorla, D., & Pugliese, R. (2005). Pattern matching over a dynamic network of tuple spaces. In M. Steffen & G. Zavattaro (Eds.), *Fmoods* (Vol. 3535, p. 1-14). Springer.

de Nicola, R., Ferrari, G. L., & Pugliese, R. (1998). Klaim: A kernel language for agents interaction and mobility. *IEEE Trans. Softw. Eng.*, *24*(5), 315–330. Retrieved from http://dx.doi.org/10.1109/32.685256 doi: 10.1109/32.685256

Dimkov, T. (2012). *Alignment of organizational security policies – theory and practice* (Unpublished doctoral dissertation). University of Twente.

Dimkov, T., Pieters, W., & Hartel, P. H. (2010, March). Portunes: representing attack scenarios spanning through the physical, digital and social domain. In *Proceedings of the Joint Workshop on Automated Reasoning for Security Protocol*

Analysis and Issues in the Theory of Security (ARSPA-WITS'10). Revised Selected Papers, Paphos, Cyprus (Vol. 6186, pp. 112–129). Berlin: Springer Verlag. http://eprints.eemcs.utwente.nl/17295/.

Dragovic, B. (2006). Casper: Containment-aware security for pervasive computing environments. *Doctor of philosophy, St John's College, University of Cambridge (March 2006)*.

Dragovic, B., & Crowcroft, J. (2004). Information exposure control through data manipulation for ubiquitous computing. In *Proceedings of the 13th New Security Paradigms Workshop* (pp. 57–64).

Dragovic, B., & Crowcroft, J. (2005). Containment: from context awareness to contextual effects awareness. In *Proceedings of 2nd International Workshop on Software Aspects of Context. CEUR Workshop Proceedings.*

Egelman, S., Brush, A. B., & Inkpen, K. M. (2008). Family accounts: a new paradigm for user accounts within the home environment. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (pp. 669–678). New York, NY, USA: ACM. doi: 10.1145/1460563.1460666

ENISA. (2009). *Cloud Computing Risk Assessment* (Tech. Rep.).

Franch, X. (2006). On the quantitative analysis of agent-oriented models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4001*, 495-509. doi: 10.1007/11767138_33

Franqueira, V. N. L., Lopes, R. H. C., & van Eck, P. (2009). Multi-step attack modelling and simulation (msams) framework based on mobile ambients. In *Proceedings of the 2009 ACM symposium on Applied Computing* (pp. 66–73). New York, NY, USA: ACM. doi: 10.1145/1529282.1529294

Gadyatskaya, O., Hansen, R. R., Larsen, K. G., Legay, A., Olesen, M. C., & Poulsen, D. B. (2016). Modelling attack-defense trees using timed automata. In M. Fränzle & N. Markey (Eds.), *Formal Modeling and Analysis of Timed Systems - 14th International Conference, FORMATS 2016, Quebec, QC, Canada, August 24-26, 2016, Proceedings* (Vol. 9884, pp. 35–50). Springer. Retrieved from http://dx.doi.org/10.1007/978-3-319-44878-7_3 doi: 10.1007/978-3-319-44878-7_3

Hay, B., Nance, K., & Bishop, M. (2011). Storm Clouds Rising: Security Challenges for IaaS Cloud Computing. In *Proceedings of the 2011 44th Hawaii International Conference on System Sciences* (pp. 1–7). Washington, DC, USA: IEEE Computer Society.

Karpati, P., Opdahl, A. L., & Sindre, G. (2011). *HARM: Hacker Attack Representation Method.* doi: 10.1007/978-3-642-29578-2_10

Kriaa, S., Bouissou, M., & Pietre-Cambacedes, L. (2012, October). Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments. In *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)* (pp. 1–8). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6378942 doi: 10.1109/CRISIS.2012.6378942

Lenzini, G., Mauw, S., & Ouchani, S. (2015). Security analysis of socio-technical physical systems. *Computers and Electrical Engineering*. Retrieved from http://www.sciencedirect.com/science/article/pii/S0045790615000671 (Available online 6 April 2015)

Maiden, N., Jones, S., Manning, S., Greenwood, J., & Renou, L. (2004). Model-driven

requirements engineering: synchronising models in an air traffic management case study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3084*, 368-383.

Mathew, S., Britt, D., Giomundo, R., Upadhyaya, S., Sudit, M., & Stotz, A. (2005). Real-time multistage attack awareness through enhanced intrusion alert clustering. In *Military Communications Conference, 2005. MILCOM 2005. IEEE* (pp. 1801–1806 Vol. 3). doi: 10.1109/MILCOM.2005.1605934

Mathew, S., Upadhyaya, S., Ha, D., & Ngo, H. (2008). Insider abuse comprehension through capability acquisition graphs. In *Information Fusion, 2008 11th International Conference on* (pp. 1–8).

Mell, P., & Grance, T. (2009a, October). *Effectively and Securely Using the Cloud Computing Paradigm.*

Mell, P., & Grance, T. (2009b, October). *The NIST Definition of Cloud Computing.*

Mohaghegh, Z. (2010). Combining system dynamics and bayesian belief networks for socio-technical risk analysis. *ISI 2010 - 2010 IEEE International Conference on Intelligence and Security Informatics: Public Safety and Security*, 196-201. doi: 10.1109/ISI.2010.5484736

Mohaghegh, Z., Kazemi, R., & Mosleh, A. (2009). Incorporating organizational factors into probabilistic risk assessment (pra) of complex socio-technical systems: A hybrid technique formalization. *Reliability Engineering and System Safety*, *94*(5), 1000-1018. (cited By 95) doi: 10.1016/j.ress.2008.11.006

Pieters, W. (2011a). Representing humans in system security models: An actor-network approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *2*(1), 75–92. http://eprints.eemcs.utwente.nl/19934/.

Pieters, W. (2011b). Security and privacy in the clouds: a bird's eye view [Technical Report]. In S. Gutwirth, Y. Poullet, P. De Hert, & R. Leenes (Eds.), *Computers, privacy and data protection: an element of choice* (pp. 445–457). Dordrecht: Springer. http://eprints.eemcs.utwente.nl/19837/.

Pieters, W., Barendse, J., Ford, M., Heath, C. P. R., Probst, C. W., & Verbij, R. (2016). The navigation metaphor in security economics. *IEEE Security & Privacy*, *14*(3), 14–21. Retrieved from http://dx.doi.org/10.1109/MSP.2016.47 doi: 10.1109/MSP.2016.47

Pieters, W., Dimkov, T., & Pavlovic, D. (2013, June). Security Policy Alignment: A Formal Approach. *IEEE Systems Journal*, *7*(2), 275–287. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6363516 doi: 10.1109/JSYST.2012.2221933

Probst, C. W., & Hansen, R. R. (2008). An extensible analysable system model. *Information Security Technical Report*, *13*(4), 235–246. Retrieved from http://dx.doi.org/10.1016/j.istr.2008.10.012 doi: 10.1016/j.istr.2008.10.012

Probst, C. W., Hansen, R. R., & Nielson, F. (2007). Where can an insider attack? In *Proceedings of the 4th International Conference on Formal Aspects in Security and Trust* (pp. 127–142). Berlin, Heidelberg: Springer-Verlag. Retrieved from http://dl.acm.org/citation.cfm?id=1777688.1777697

Probst, C. W., & Hunker, J. (2009). The risk of risk analysis—and its relation to the economics of insider threats. In *Proceedings of the 8th Annual Workshop on the Economics of Information Security (WEIS 2009).*

Probst, C. W., Kammüller, F., & Hansen, R. R. (2016). Formal modelling and analysis of socio-technical systems. In C. W. Probst, C. Hankin, & R. R. Hansen (Eds.), *Semantics, logics, and calculi* (Vol. 9560, pp. 54–73). Springer. Retrieved from http://dx.doi.org/10.1007/978-3-319-27810-0_3 doi: 10.1007/978-3-319-27810-0_3

Probst, C. W., Willemson, J., & Pieters, W. (2016). The attack navigator. In S. Mauw, B. Kordy, & S. Jajodia (Eds.), *Graphical Models for Security - Second International Workshop, GraMSec 2015, Verona, Italy, July 13, 2015, Revised Selected Papers* (Vol. 9390, pp. 1–17). Springer. Retrieved from http://dx.doi.org/10.1007/978-3-319-29968-6_1 doi: 10.1007/978-3-319-29968-6_1

Samarji, L., Cuppens, F., Cuppens-Boulahia, N., Kanoun, W., & Dubus, S. (2013). Situation calculus and graph based defensive modeling of simultaneous attacks. In *Cyberspace Safety and Security* (pp. 132–150). Springer.

Santhi, N., Yan, G., & Eidenbenz, S. (2010). Cybersim: Geographic, temporal, and organizational dynamics of malware propagation. *Proceedings - Winter Simulation Conference*, 2876-2887. doi: 10.1109/WSC.2010.5678982

Sarkar, A., Kohler, S., Riddle, S., Ludäscher, B., & Bishop, M. (2014). Insider attack identification and prevention using a declarative approach. In *Security and Privacy Workshops (SPW), 2014 IEEE* (pp. 265–276).

Scott, D. (2004). *Abstracting Application-Level Security Policy for Ubiquitous Computing* (Doctoral dissertation, University of Cambridge). Retrieved from http://www.cl.cam.ac.uk/research/dtg/www/files/publications/public/djs55/approved.pdf (PhD Thesis)

Shahriari, H. R., Makarem, M. S., Sirjani, M., Jalili, R., & Movaghar, A. (2010, September). Vulnerability analysis of networks to detect multiphase attacks using the actor-based language Rebeca. *Computers & Electrical Engineering*, *36*(5), 874–885. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0045790608000451 doi: 10.1016/j.compeleceng.2008.04.009

Sommestad, T., Ekstedt, M., & Holm, H. (2012). The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *Systems Journal, IEEE*, *PP*(99), 1–1. doi: 10.1109/JSYST.2012.2221853

Sommestad, T., Ekstedt, M., & Johnson, P. (2010). A probabilistic relational model for security risk analysis. *Computers & Security*, *29*(6), 659 - 679. doi: http://dx.doi.org/10.1016/j.cose.2010.02.002

Ten, C.-W., Manimaran, G., & Liu, C.-C. (2010, July). Cybersecurity for Critical Infrastructures: Attack and Defense Modeling. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *40*(4), 853–865. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5477189 doi: 10.1109/TSMCA.2010.2048028

The Open Group. (2013). *ArchiMate® 2.1 specification*. Van Haren Publishing.

The TRESPASS Project. (2016a). *Xml files for the case studies.* (Available from https://www.trespass-project.eu/Models)

The TRESPASS Project. (2016b). *Xml schema definition for model files.* (Available from https://www.trespass-project.eu/schemas/TREsPASS_model.xsd)

The TRESPASS Project. (2016c). *Xml schema definition for scenario files.* (Available from https://www.trespass-project.eu/schemas/TREsPASS_scenario.xsd)

The TRESPASS Project, D1.1.2. (2015). *Final specifications and requirements for socio-technical security models.* (Deliverable D1.1.2)

The TRESPASS Project, D1.2.2. (2015). *Final policy-specification language.* (Deliverable D1.2.2)

The TRESPASS Project, D1.3.2. (2015). *Extensibility of socio-technical security models.* (Deliverable D1.3.2)

The TRESPASS Project, D2.4.1. (2016). *TRESPASS information system.* (Deliverable D2.4.1)

The TRESPASS Project, D3.2.1. (2015). *TRESPASS extraction methods for stochastic models.* (Deliverable D3.2.1)

The TRESPASS Project, D3.4.1. (2014). *Attack generation from socio-technical security models.* (Deliverable D3.4.1)

The TRESPASS Project, D3.4.2. (2016). *Methods for attack generation, preventive measures, and ranking.* (Deliverable D3.4.2)

The TRESPASS Project, D4.2.2. (2016). *Methods for visualization of information security risks.* (Deliverable D4.2.2)

Vintr, Z., Valis, D., & Malach, J. (2012, October). Attack tree-based evaluation of physical protection systems vulnerability. In *2012 ieee international carnahan conference on security technology (iccst)* (pp. 59–65). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6393538 doi: 10.1109/CCST.2012.6393538

Xie, A., Chen, G., Wang, Y., Chen, Z., & Hu, J. (2009, July). A New Method to Generate Attack Graphs. In *2009 third ieee international conference on secure software integration and reliability improvement* (pp. 401–406). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5325344 doi: 10.1109/SSIRI.2009.32

Zhao, F., Huang, H., Jin, H., & Zhang, Q. (2011). A hybrid ranking approach to estimate vulnerability for dynamic attacks. *Computers and Mathematics with Applications*, *62*(12), 4308–4321. doi: j.camwa.2011.09.031

A. The TRESPASS Model and Scenario XML Structure

A.1. XML Schema for TRESPASS Model

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    targetNamespace="https://www.trespass-project.eu/schemas/TREsPASS_model"
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_model"
    >
  <!-- ================================================================ -->
  <!-- system is the root node                                          -->
  <!-- ================================================================ -->
  <xs:element name="system">
    <xs:complexType>
      <xs:all>
        <xs:element name="title" />
        <xs:element minOccurs="0" name="locations" type="locationType" />
        <xs:element minOccurs="0" name="edges" type="edgeType" />
        <xs:element minOccurs="0" name="assets" type="assetType" />
        <xs:element minOccurs="0" name="actors" type="actorsType" />
        <xs:element minOccurs="0" name="policies" type="policyType" />
        <xs:element minOccurs="0" name="processes" type="processType" />
        <xs:element minOccurs="0" name="predicates" type="predicatesType" />
        <xs:element minOccurs="0" name="metrics" type="metricsType" />
      </xs:all>
      <xs:attribute name="author" use="required" />
      <xs:attribute name="date" use="required" />
      <xs:attribute name="version" use="required" />
      <xs:attribute name="id" use="required" />
      <xs:attribute name="anm_data" use="optional" />
    </xs:complexType>
  </xs:element>
  <!-- ================================================================ -->
  <!-- locations have an id and some attributes                         -->
  <!-- ================================================================ -->
  <xs:complexType name="locationType" >
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="location" >
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="1">
            <xs:element name="atLocations" type="xs:IDREFS" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required" />
          <xs:attribute name="type" type="xs:string" use="optional" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="locationDomain">
    <xs:restriction base="xs:string">
      <xs:enumeration value="physical" />
      <xs:enumeration value="cyber" />
      <xs:enumeration value="other" />
    </xs:restriction>
  </xs:simpleType>
  <!-- ================================================================ -->
  <!-- edges have a src and a tgt and a kind                            -->
  <!-- edges are directed by default                                    -->
  <!-- ================================================================ -->
  <xs:complexType name="edgeType" >
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="edge" >
        <xs:complexType>
          <xs:sequence>
            <xs:element name="source" type="xs:IDREF" />
            <xs:element name="target" type="xs:IDREF" />
          </xs:sequence>
```

```
        <xs:attribute name="directed" type="xs:boolean" default="true" />
        <xs:attribute name="kind" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<!-- ================================================================ -->
<!-- assets are items or data                                         -->
<!-- ================================================================ -->
<xs:complexType name="assetType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="item" type="itemType" />
      <xs:element name="data" type="dataType" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!-- ================================================================ -->
<!-- items have a name and id and are located at other locations      -->
<!-- ================================================================ -->
<xs:complexType name="itemType" >
  <xs:sequence>
    <xs:element name="atLocations" type="xs:IDREFS" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="type" type="xs:string" use="optional" />
</xs:complexType>
<!-- ================================================================ -->
<!-- data has a name and an id and is located at other locations      -->
<!-- data also has a value                                            -->
<!-- ================================================================ -->
<xs:complexType name="dataType" >
  <xs:sequence>
    <xs:element name="atLocations" type="xs:IDREFS" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="value" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" use="optional" />
</xs:complexType>
<!-- ================================================================ -->
<!-- actors have an id and are located at other locations             -->
<!-- ================================================================ -->
<xs:complexType name="actorsType" >
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="actor" type="actorType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="actorType" >
  <xs:sequence>
    <xs:element name="atLocations" type="xs:IDREFS" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="type" type="xs:string" use="optional" />
</xs:complexType>
<!-- ================================================================ -->
<!-- policies have set of credentials and set of enabled actions      -->
<!-- ================================================================ -->
<xs:complexType name="policyType" >
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded" name="policy" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="credentials" >
            <xs:complexType>
              <xs:sequence minOccurs="0" maxOccurs="unbounded" >
                <xs:choice>
                  <xs:element name="credLocation" type ="credLocationType" />
                  <xs:element name="credItem" type="credItemType" />
                  <xs:element name="credData" type="credDataType" />
                  <xs:element name="credPredicate" type="credPredicateType" />
                </xs:choice>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="enabled" type="actionsType" />
          <xs:element name="atLocations" type="xs:IDREFS" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<!-- ================================================================ -->
<!-- credential elements                                              -->
<!-- ================================================================ -->
<xs:complexType name="credLocationType" >
```

```
  <xs:attribute name="id" type="xs:IDREF" use="required" />
</xs:complexType>
<xs:complexType name="credDataType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="value" type="xs:string" />
      <xs:element name="variable" type="xs:string" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="credItemType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="credData" type="credDataType" />
      <xs:element name="credItem" type="credItemType" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="credPredicateType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="variable" type="xs:string" />
      <xs:element name="value" type="xs:string" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" use="required" />
</xs:complexType>
<!-- ------------------------------------------------------------ -->
<!-- processes                                                    -->
<!-- ------------------------------------------------------------ -->
<xs:complexType name="processType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:element name="process" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="actions" type="actionsType" />
          <xs:element name="atLocations" type="xs:IDREFS" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required" />
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="actionsType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="in" type="inOutType" />
      <xs:element name="out" type="inOutType" />
      <xs:element name="move" />
      <xs:element name="eval" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="inOutType" >
  <xs:sequence>
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:choice>
        <xs:element name="value" type="xs:string" />
        <xs:element name="wildcard" />
        <xs:element name="variable" type="xs:string" />
        <xs:element name="input" type="xs:string" />
        <xs:element name="tuple" type="tupleType" />
      </xs:choice>
    </xs:sequence>
    <xs:sequence minOccurs="0" maxOccurs="1" >
      <xs:choice>
        <xs:element name="locvar" type="xs:string" />
        <xs:element name="locval" type="xs:string" />
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="logged" type="xs:boolean" use="optional"/>
</xs:complexType>
<xs:complexType name="tupleType" >
  <xs:sequence minOccurs="0" maxOccurs="unbounded" >
    <xs:choice>
      <xs:element name="value" type="xs:string" />
      <xs:element name="wildcard" />
      <xs:element name="variable" type="xs:string" />
      <xs:element name="input" type="xs:string" />
      <xs:element name="tuple" type="tupleType" />
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!-- ------------------------------------------------------------ -->
<!-- predicates specify properties of locations, actors, etc      -->
```

```
  <!-------------------------------------------------------------->
  <xs:simpleType name="stringlist">
    <xs:list itemType="xs:string" />
  </xs:simpleType>
  <xs:complexType name="predicatesType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:element name="predicate" >
        <xs:complexType>
          <xs:sequence minOccurs="1" maxOccurs="unbounded" >
            <xs:element name="value" type="stringlist" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required" />
          <xs:attribute name="arity" type="xs:int" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <!-------------------------------------------------------------->
  <!-- metrics specify numbers for a name and an optional namespace -->
  <!-------------------------------------------------------------->
  <xs:complexType name="metricsType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:element name="metric" >
        <xs:complexType>
          <xs:attribute name="element" type=''xs:IDREF'' use=''required''/>
          <xs:attribute name="name" use="required" />
          <xs:attribute name="value" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

A.2. XML Schema for TRESPASS Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    targetNamespace="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    >
  <xs:element name="scenario">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="model" type="xs:string"/>
        <xs:choice>
          <xs:element name="actionGoal"  type="actionGoal"/>
          <xs:element name="locationGoal"  type="locationGoal"/>
          <xs:element name="assetGoal" type="assetGoal"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="author" use="required"/>
      <xs:attribute form="unqualified" name="date" use="required"/>
      <xs:attribute name="version" use="required"/>
    </xs:complexType>
  </xs:element>
  <!-- ================================================================ -->
  <!-- asset goal type                                                  -->
  <!-- specifies data and location                                      -->
  <!-- ================================================================ -->
  <xs:complexType name="assetGoal">
    <xs:sequence minOccurs="1" maxOccurs="1">
      <xs:element name="asset" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="attacker" type="xs:string" use="required"/>
    <xs:attribute name="profit" type="xs:decimal" use="required"/>
  </xs:complexType>
  <!-- ================================================================ -->
  <!-- location goal type                                               -->
  <!-- specifies data and location                                      -->
  <!-- ================================================================ -->
  <xs:complexType name="locationGoal">
    <xs:sequence>
      <xs:sequence minOccurs="1">
        <xs:element name="asset" type="assetType" />
      </xs:sequence>
      <xs:element name="location" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="attacker" type="xs:string" use="required"/>
    <xs:attribute name="profit" type="xs:decimal" use="required"/>
  </xs:complexType>
  <xs:complexType name="assetType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:choice>
        <xs:element name="value" type="xs:string" />
        <xs:element name="tuple" type="assetTupleType" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="assetTupleType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:choice>
        <xs:element name="value" type="xs:string" />
        <xs:element name="tuple" type="assetTupleType" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <!-- ================================================================ -->
  <!-- action goal type                                                 -->
  <!-- specifies data and location                                      -->
  <!-- ================================================================ -->
  <xs:complexType name="actionGoal">
    <xs:sequence>
      <xs:element name="credentials">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
              <xs:element name="credLocation" type ="credLocation" />
              <xs:element name="credItem" type="credItem" />
              <xs:element name="credData" type="credData" />
              <xs:element name="credPredicate" type="credPredicate"/>
            </xs:choice>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="enabled" type="actionsType" />
    </xs:sequence>
```

```
    <xs:attribute name="attacker" type="xs:string" use="required"/>
    <xs:attribute name="profit" type="xs:decimal" use="required"/>
  </xs:complexType>
  <xs:complexType name="actionsType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:choice>
        <xs:element name="in" type="inOutType" />
        <xs:element name="out" type="inOutType" />
        <xs:element name="move" />
        <xs:element name="eval" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="inOutType" >
    <xs:sequence>
      <xs:sequence minOccurs="0" maxOccurs="unbounded" >
        <xs:choice>
          <xs:element name="value" type="xs:string" />
          <xs:element name="wildcard" />
          <xs:element name="variable" type="xs:string" />
          <xs:element name="input" type="xs:string" />
          <xs:element name="tuple" type="tupleType" />
        </xs:choice>
      </xs:sequence>
      <xs:sequence minOccurs="0" maxOccurs="1" >
        <xs:choice>
          <xs:element name="locvar" type="xs:string" />
          <xs:element name="locvalue" type="xs:string" />
        </xs:choice>
      </xs:sequence>
    </xs:sequence>
    <xs:attribute name="logged" type="xs:boolean" use="optional"/>
  </xs:complexType>
  <xs:complexType name="tupleType" >
    <xs:sequence minOccurs="0" maxOccurs="unbounded" >
      <xs:choice>
        <xs:element name="value" type="xs:string" />
        <xs:element name="wildcard" />
        <xs:element name="variable" type="xs:string" />
        <xs:element name="input" type="xs:string" />
        <xs:element name="tuple" type="tupleType" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType      name="credLocation">
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="credData">
    <xs:choice>
      <xs:element name="value" type="xs:string" />
      <xs:element name="variable" type="xs:string" />
    </xs:choice>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="credItem">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element name="credData" type="credData" />
        <xs:element name="credItem" type="credItem" />
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
  </xs:complexType>
  <xs:complexType name="credPredicate">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element name="variable" type="xs:string"/>
        <xs:element name="value" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" use="required"/>
  </xs:complexType>
</xs:schema>
```

B. The Case Study Models

B.1. IPTV

B.1.1. Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_scenario.xsd"
    id="IPTV Scenario"
    author="CW Probst"
    date="2016-10-16 19:15:00"
    version="1.0">
   <model>IPTV-Model.xml</model>
   <actionGoal attacker="Z" profit="1000">
      <credentials>
                <credPredicate name="role">
                    <value>Employee</value>
                                <variable>Z</variable>
                </credPredicate>
                <credItem name="Card">
                                <credData name="Owner">
                                        <variable>O</variable>
                                </credData>
                </credItem>
                <credPredicate name="role">
                    <value>Customer</value>
                                <variable>O</variable>
                </credPredicate>
                <credPredicate name="isActor">
                                <variable>Z</variable>
                </credPredicate>
      </credentials>
      <enabled>
         <in />
      </enabled>
   </actionGoal>
</scenario>
```

B.1.2. Model

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<system
  xmlns="https://www.trespass-project.eu/schemas/TREsPASS_model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_model
                      https://www.trespass-project.eu/schemas/TREsPASS_model.xsd"
  author="Christian W Probst"
  version="1.0.0"
  date="2016-10-10 10:02:05"
  id="IPTV Model">
  <title>IPTV Model</title>
  <actors>
    <actor id="Fred" name="Fred"><atLocations>city</atLocations></actor>
    <actor id="Margarethe" name="Margarethe"><atLocations>home</atLocations></actor>
    <actor id="Charlie" name="Charlie"><atLocations>city</atLocations></actor>
    <actor id="Fritz" name="Fritz"><atLocations>home</atLocations></actor>
  </actors>
  <edges>
    <edge directed="false">
      <source>home</source>
      <target>door</target>
    </edge>
    <edge directed="false">
      <source>door</source>
```

```
      <target>city</target>
    </edge>
    <edge directed="false">
      <source>city</source>
      <target>atm</target>
    </edge>
    <edge directed="false">
      <source>bank</source>
      <target>city</target>
    </edge>
    <edge directed="false">
      <source>remote</source>
      <target>settop</target>
    </edge>
    <edge directed="false">
      <source>settop</source>
      <target>server</target>
    </edge>
  </edges>
  <locations>
    <location id="home"   name="Home"/>
    <location id="city"   name="City"/>
    <location id="door"   name="Door"/>
    <location id="atm"   name="ATM"/>
    <location id="bank"   name="Bank"/>
  </locations>
  <policies>
    <policy id="p001">
      <credentials>
        <credPredicate name="trusts"><value>Margarethe</value><variable>X</variable></credPredicate>
      </credentials>
      <enabled>
        <move />
      </enabled>
      <atLocations>door</atLocations>
    </policy>
    <policy id="p002">
      <credentials>
        <credItem name="Card"><credData name="PIN"><variable>Y</variable></credData></credItem>
        <credData name="PIN"><variable>Y</variable></credData>
      </credentials>
      <enabled>
        <in />
      </enabled>
      <atLocations>safe_box</atLocations>
    </policy>
   <policy id="p003">
     <credentials>
       <credLocation name="home"/>
     </credentials>
     <enabled><out /></enabled>
     <atLocations>remote</atLocations>
   </policy>
   <policy id="p004">
     <credentials>
       <credLocation name="Remote"/>
     </credentials>
     <enabled><out /></enabled>
     <atLocations>remote</atLocations>
   </policy>
   <policy id="p005">
     <credentials>
       <credPredicate name="technician"><variable>X</variable></credPredicate>
     </credentials>
     <enabled><out><value>Firmware</value><wildcard /></out></enabled>
     <atLocations>settop</atLocations>
   </policy>
   <policy id="p006">
     <credentials>
       <credLocation name="server" />
     </credentials>
     <enabled>
       <out><value>transfer</value><wildcard /><wildcard /><wildcard /></out>
     </enabled>
     <atLocations>account</atLocations>
   </policy>
   <policy id="p007">
     <credentials>
       <credLocation name="server" />
     </credentials>
     <enabled>
       <out><value>deposit</value><wildcard /><wildcard /></out>
     </enabled>
     <atLocations>account</atLocations>
   </policy>
   <policy id="p008">
     <credentials>
       <credLocation name="settop"/>
```

```
      </credentials>
      <enabled><out /></enabled>
      <atLocations>server</atLocations>
    </policy>
  </policies>
  <predicates>
    <predicate id="trusts" arity="2">
      <value>Margarethe Charlie</value>
    </predicate>
    <predicate id="role" arity="2">
      <value>Employee Fred</value>
      <value>Employee Charlie</value>
      <value>Customer Margarethe</value>
      <value>Technician Fritz</value>
    </predicate>
  </predicates>
  <assets>
    <data id="owner" name="Owner" value="Margarethe" >
      <atLocations>card</atLocations>
    </data>
    <data id="pin" name="PIN" value="1" >
      <atLocations>Margarethe card</atLocations>
    </data>
    <data id="password" name="Password" value="2" >
      <atLocations>Margarethe</atLocations>
    </data>
    <data id="account" name="account" value="1234" >
      <atLocations>card</atLocations>
    </data>
    <data id="number" name="number" value="1234" >
      <atLocations>account</atLocations>
    </data>
    <data id="password-2" name="Password" value="2" >
      <atLocations>settop</atLocations>
    </data>
    <data id="password-3" name="Password" value="2" >
      <atLocations>account</atLocations>
    </data>
    <item id="money" name="Money" >
      <atLocations>safe-box</atLocations>
    </item>
    <item id="card" name="Card" >
      <atLocations>Margarethe</atLocations>
    </item>
    <item id="settop" name="Settop" >
      <atLocations>home</atLocations>
    </item>
    <item id="remote" name="Remote" >
      <atLocations>home</atLocations>
    </item>
    <item id="server" name="Server" >
      <atLocations>bank</atLocations>
    </item>
    <item id="safe-box" name="SafeBox" >
      <atLocations>atm</atLocations>
    </item>
    <item id="account" name="account" >
      <atLocations>server</atLocations>
    </item>
    <item id="money-2" name="Money" >
      <atLocations>account</atLocations>
    </item>
  </assets>
<processes>
    <process id="P_remote">
        <atLocations>remote</atLocations>
        <actions>
            <in><value>card</value><input>card</input></in>
            <in><value>pin</value><input>userpin</input></in>
            <in><value>password</value><input>pwd</input></in>
            <in><value>amount</value><input>amount</input></in>
            <in><value>account</value><input>account</input></in>
            <in><value>pin</value><variable>userpin</variable><locvar>card</locvar></in>
            <in><value>account</value><variable>ownaccount</variable><locvar>card</locvar></in>
            <out><value>transfer</value><variable>pwd</variable>
                        <variable>amount</variable><variable>ownaccount</variable><variable>account</variable>
                        <locval>settop</locval></out>
        </actions>
    </process>
    <process id="P_settop">
            <in><value>transfer</value><input>pwd</input>
                        <input>amount</input><input>ownaccount</input><input>account</input>
                        <locval>settop</locval></in>
            <in><value>password</value><variable>pwd</variable></in>
            <out><value>transfer</value><variable>pwd</variable>
                        <variable>amount</variable><variable>ownaccount</variable><variable>account</variable>
                        <locval>settop</locval></out>
    </process>
```

```
    <process id="P_server">
            <in><value>transfer</value><input>pwd</input>
                    <input>amount</input><input>ownaccount</input><input>account</input></in>
            <out><value>transfer</value><variable>account</variable>
                    <variable>pwd</variable><variable>amount</variable><locvar>ownaccount</locvar></out>
    </process>
  </processes>
</system>
```

B.2. Telecommunication

B.2.1. Scenario

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<scenario
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_scenario
                        https://www.trespass-project.eu/schemas/TREsPASS_scenario.xsd"
    author="Christian W Probst"
    version="1.0"
    date="2016-10-15 17:21:00"
    id="PBX_scenario">
  <locationGoal attacker="actor__attacker" profit="5000">
    <asset>
      <tuple><value>ring</value><value>internalPhone</value><value>phoneB</value></tuple>
      <tuple><value>ring</value><value>laptop</value><value>phoneB</value></tuple>
      <tuple><value>ring</value><value>PBX</value><value>phoneB</value></tuple>
    </asset>
    <location>phoneB</location>
  </locationGoal>
  <model>TELCO-Model.xml</model>
</scenario>
```

B.2.2. Model

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<system
  xmlns="https://www.trespass-project.eu/schemas/TREsPASS_model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_model
                      https://www.trespass-project.eu/schemas/TREsPASS_model.xsd"
  author="Christian W Probst"
  version="1.0.0"
  date="2016-10-11 15:52:11" >
  <title>TELCO Model</title>
  <actors>
    <actor id="john"  name="john"><atLocations>world</atLocations></actor>
    <actor id="admin"  name="admin"><atLocations>world</atLocations></actor>
    <actor id="attacker"  name="attacker"><atLocations>world</atLocations></actor>
  </actors>
  <locations>
    <location id="johnsOffice"  name="johnsOffice"/>
    <location id="serverRoom"  name="serverRoom"/>
    <location id="reception"  name="reception"/>
    <location id="doorOffice"  name="doorOffice"/>
    <location id="doorServer"  name="doorServer"/>
    <location id="world"  name="world"/>
    <location id="TSPA"  name="TSPA"/>
    <location id="TSPB"  name="TSPB"/>
  </locations>
  <edges>
    <edge directed="false"><source>reception</source><target>doorOffice</target></edge>
    <edge directed="false"><source>reception</source><target>doorserver</target></edge>
    <edge directed="false"><source>doorOffice</source><target>johnsOffice</target></edge>
    <edge directed="false"><source>doorServer</source><target>serverRoom</target></edge>
    <edge directed="false"><source>world</source><target>reception</target></edge>
    <edge directed="false"><source>TSPB</source><target>world</target></edge>
    <edge directed="false"><source>TSPA</source><target>world</target></edge>
    <edge directed="false" kind="network"><source>internalPhone</source><target>PBX</target></edge>
    <edge directed="false" kind="network"><source>laptop</source><target>PBX</target></edge>
    <edge directed="false" kind="network"><source>adminPhone</source><target>PBX</target></edge>
    <edge directed="false" kind="network"><source>PBX</source><target>serverA</target></edge>
    <edge directed="false" kind="network"><source>serverA</source><target>serverB</target></edge>
    <edge directed="false" kind="network"><source>serverB</source><target>phoneB</target></edge>
    <edge directed="false" kind="network"><source>serverA</source><target>phoneA</target></edge>
  </edges>
  <assets>
    <data id="password" name="password" value="john" ><atLocations>john laptop</atLocations></data>
    <data id="login" name="login" value="john" ><atLocations>john laptop</atLocations></data>
    <data id="credential" name="credential" value="john" ><atLocations>john internalPhone laptop</atLocations></data>
    <data id="username" name="username" value="john" ><atLocations>john internalPhone laptop</atLocations></data>
    <data id="credential-2" name="credential" value="admin" ><atLocations>PBX admin adminPhone</atLocations></data>
    <data id="username-2" name="username" value="admin" ><atLocations>admin PBX adminPhone</atLocations></data>
    <item id="key" name="key" ><atLocations>john</atLocations></item>
    <item id="card" name="AccessCard" ><atLocations>john</atLocations></item>
    <data id="pin" name="PIN" value="42" ><atLocations>card john</atLocations></data>
    <item id="card-2" name="AccessCard" ><atLocations>admin</atLocations></item>
    <data id="pin-2" name="PIN" value="43" ><atLocations>card-2 admin</atLocations></data>
    <item id="PBX" name="PBX" ><atLocations>serverRoom</atLocations></item>
```

```
    <item id="internalPhone" name="internalPhone" ><atLocations>johnsOffice</atLocations></item>
    <item id="laptop" name="laptop" ><atLocations>johnsOffice</atLocations></item>
    <item id="phoneA" name="phoneA" ><atLocations>world</atLocations></item>
    <item id="phoneB" name="phoneB" ><atLocations>world</atLocations></item>
    <item id="adminPhone" name="adminPhone" ><atLocations>world</atLocations></item>
    <item id="serverA" name="serverA" ><atLocations>TSPA</atLocations></item>
    <item id="serverB" name="serverB" ><atLocations>TSPB</atLocations></item>
    <item name="routingtable" id="PBX_routingtable"><atLocations>PBX</atLocations></item>
    <item name="routingtable" id="serverA_routingtable"><atLocations>serverA</atLocations></item>
    <item name="routingtable" id="serverB_routingtable"><atLocations>serverB</atLocations></item>
    <data id="routePBX1" name="adminPhone" value="adminPhone"><atLocations>PBX_routingtable</atLocations></data>
    <data id="routePBX2" name="internalPhone" value="internalPhone"><atLocations>PBX_routingtable</atLocations></data>
    <data id="routePBX3" name="phoneA" value="serverA"><atLocations>PBX_routingtable</atLocations></data>
    <data id="routePBX4" name="phoneB" value="serverA"><atLocations>PBX_routingtable</atLocations></data>
    <data id="routeserverA1" name="internalPhone" value="PBX"><atLocations>serverA_routingtable</atLocations></data>
    <data id="routeserverA2" name="adminPhone" value="PBX"><atLocations>serverA_routingtable</atLocations></data>
    <data id="routeserverA3" name="phoneA" value="phoneA"><atLocations>serverA_routingtable</atLocations></data>
    <data id="routeserverA4" name="phoneB" value="serverB"><atLocations>serverA_routingtable</atLocations></data>
    <data id="routeserverB1" name="internalPhone" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
    <data id="routeserverB2" name="adminPhone" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
    <data id="routeserverB3" name="phoneA" value="serverA"><atLocations>serverB_routingtable</atLocations></data>
    <data id="routeserverB4" name="phoneB" value="phoneB"><atLocations>serverB_routingtable</atLocations></data>
  </assets>
  <policies>
    <policy id="johnsOffice_door">
      <atLocations>doorOffice</atLocations>
      <credentials><credItem name="key"/></credentials>
      <enabled><move/></enabled>
    </policy>
    <policy id="phone_call">
      <atLocations>internalPhone</atLocations>
      <credentials>
        <credData name="credential"><value>john</value></credData>
        <credData name="username"><value>john</value></credData>
      </credentials>
      <enabled>
        <out><tuple><value>call</value><variable>callee</variable></tuple></out>
      </enabled>
    </policy>
    <policy id="laptop_call">
      <atLocations>laptop</atLocations>
      <credentials>
        <credData name="password"><value>john</value></credData>
        <credData name="login"><value>john</value></credData>
        <credData name="credential"><value>john</value></credData>
        <credData name="username"><value>john</value></credData>
      </credentials>
      <enabled>
        <out><tuple><value>call</value><variable>callee</variable></tuple></out>
      </enabled>
    </policy>
    <policy id="PBX_call">
      <atLocations>PBX</atLocations>
      <credentials>
        <credData name="credential"><value>admin</value></credData>
        <credData name="username"><value>admin</value></credData>
      </credentials>
      <enabled>
        <out><tuple><value>call</value><variable>callee</variable></tuple></out>
      </enabled>
    </policy>
    <policy id="serverRoom_door">
      <atLocations>doorServer</atLocations>
      <credentials>
        <credItem name="AccessCard"><credData name="PIN"><variable>Y</variable></credData></credItem>
        <credData name="PIN"><variable>Y</variable></credData>
      </credentials>
      <enabled>
        <move />
      </enabled>
    </policy>
  </policies>
  <processes>
    <process id="P_internalPhone_call">
      <actions>
        <in><tuple><value>call</value><input>callee</input></tuple></in>
        <out><value>IP</value><value>internalPhone</value><variable>callee</variable>
            <tuple><value>ring</value><value>internalPhone</value><variable>callee</variable></tuple></out>
      </actions>
      <atLocations>internalPhone</atLocations>
    </process>
    <process id="P_internalPhone_answer">
      <actions>
        <in><tuple><value>call</value><value>internalPhone</value><input>caller</input></tuple></in>
        <out><value>IP</value><value>internalPhone</value><variable>caller</variable>
            <tuple><value>answer</value><value>internalPhone</value><variable>caller</variable></tuple></out>
      </actions>
      <atLocations>internalPhone</atLocations>
```

```
</process>
<process id="P_internalPhone_IPout">
  <actions>
    <in><value>IP</value><value>internalPhone</value><input>target</input><input>tuple</input></in>
    <out><value>IP</value><value>internalPhone</value><variable>target</variable>
        <variable>tuple</variable><locval>PBX</locval></out>
  </actions>
  <atLocations>internalPhone</atLocations>
</process>
<process id="P_internalPhone_IPin">
  <actions>
    <in><value>IP</value><input>source</input><value>internalPhone</value><input>tuple</input></in>
    <out><variable>tuple</variable></out>
  </actions>
  <atLocations>internalPhone</atLocations>
</process>
<process id="P_adminPhone_call">
  <actions>
    <in><tuple><value>call</value><input>callee</input></tuple></in>
    <out><value>IP</value><value>adminPhone</value><variable>callee</variable>
        <tuple><value>ring</value><value>adminPhone</value><variable>callee</variable></tuple></out>
  </actions>
  <atLocations>adminPhone</atLocations>
</process>
<process id="P_adminPhone_answer">
  <actions>
    <in><tuple><value>call</value><value>adminPhone</value><input>caller</input></tuple></in>
    <out><value>IP</value><value>adminPhone</value><variable>caller</variable>
        <tuple><value>answer</value><value>adminPhone</value><variable>caller</variable></tuple></out>
  </actions>
  <atLocations>adminPhone</atLocations>
</process>
<process id="P_adminPhone_IPout">
  <actions>
    <in><value>IP</value><value>adminPhone</value><input>target</input><input>tuple</input></in>
    <out><value>IP</value><value>adminPhone</value><variable>target</variable>
        <variable>tuple</variable><locval>PBX</locval></out>
  </actions>
  <atLocations>adminPhone</atLocations>
</process>
<process id="P_adminPhone_IPin">
  <actions>
    <in><value>IP</value><input>source</input><value>adminPhone</value><input>tuple</input></in>
    <out><variable>tuple</variable></out>
  </actions>
  <atLocations>adminPhone</atLocations>
</process>
<process id="P_phoneA_call">
  <actions>
    <in><tuple><value>call</value><input>callee</input></tuple></in>
    <out><value>IP</value><value>phoneA</value><variable>callee</variable>
        <tuple><value>ring</value><value>phoneA</value><variable>callee</variable></tuple></out>
  </actions>
  <atLocations>phoneA</atLocations>
</process>
<process id="P_phoneA_answer">
  <actions>
    <in><tuple><value>call</value><value>phoneA</value><input>caller</input></tuple></in>
    <out><value>IP</value><value>phoneA</value><variable>caller</variable>
        <tuple><value>answer</value><value>phoneA</value><variable>caller</variable></tuple></out>
  </actions>
  <atLocations>phoneA</atLocations>
</process>
<process id="P_phoneA_IPout">
  <actions>
    <in><value>IP</value><value>phoneA</value><input>target</input><input>tuple</input></in>
    <out><value>IP</value><value>phoneA</value><variable>target</variable>
        <variable>tuple</variable><locval>serverA</locval></out>
  </actions>
  <atLocations>phoneA</atLocations>
</process>
<process id="P_phoneA_IPin">
  <actions>
    <in><value>IP</value><input>source</input><value>phoneA</value><input>tuple</input></in>
    <out><variable>tuple</variable></out>
  </actions>
  <atLocations>phoneA</atLocations>
</process>
<process id="P_phoneB_call">
  <actions>
    <in><tuple><value>call</value><input>callee</input></tuple></in>
    <out><value>IP</value><value>phoneB</value><variable>callee</variable>
        <tuple><value>ring</value><value>phoneB</value><variable>callee</variable></tuple></out>
  </actions>
  <atLocations>phoneB</atLocations>
</process>
<process id="P_phoneB_answer">
  <actions>
```

```
      <in><tuple><value>call</value><value>phoneB</value><input>caller</input></tuple></in>
      <out><value>IP</value><value>phoneB</value><variable>caller</variable>
          <tuple><value>answer</value><value>phoneB</value><variable>caller</variable></tuple></out>
    </actions>
    <atLocations>phoneB</atLocations>
  </process>
  <process id="P_phoneB_IPout">
    <actions>
      <in><value>IP</value><value>phoneB</value><input>target</input><input>tuple</input></in>
      <out><value>IP</value><value>phoneB</value><variable>target</variable>
          <variable>tuple</variable><locval>serverB</locval></out>
    </actions>
    <atLocations>phoneB</atLocations>
  </process>
  <process id="P_phoneB_IPin">
    <actions>
      <in><value>IP</value><input>source</input><value>phoneB</value><input>tuple</input></in>
      <out><variable>tuple</variable></out>
    </actions>
    <atLocations>phoneB</atLocations>
  </process>
  <process id="P_laptop_call">
    <actions>
      <in><tuple><value>call</value><input>callee</input></tuple></in>
      <out><value>IP</value><value>laptop</value><variable>callee</variable>
          <tuple><value>ring</value><value>laptop</value><variable>callee</variable></tuple></out>
    </actions>
    <atLocations>laptop</atLocations>
  </process>
  <process id="P_laptop_answer">
    <actions>
      <in><tuple><value>call</value><value>laptop</value><input>caller</input></tuple></in>
      <out><value>IP</value><value>laptop</value><variable>caller</variable>
          <tuple><value>answer</value><value>laptop</value><variable>caller</variable></tuple></out>
    </actions>
    <atLocations>laptop</atLocations>
  </process>
  <process id="P_laptop_IPout">
    <actions>
      <in><value>IP</value><value>laptop</value><input>target</input><input>tuple</input></in>
      <out><value>IP</value><value>laptop</value><variable>target</variable>
          <variable>tuple</variable><locval>PBX</locval></out>
    </actions>
    <atLocations>laptop</atLocations>
  </process>
  <process id="P_laptop_IPin">
    <actions>
      <in><value>IP</value><input>source</input><value>laptop</value><input>tuple</input></in>
      <out><variable>tuple</variable></out>
    </actions>
    <atLocations>laptop</atLocations>
  </process>
  <process id="P_PBX_call">
    <actions>
      <in><tuple><value>call</value><input>callee</input></tuple></in>
      <out><value>IP</value><value>PBX</value><variable>callee</variable>
          <tuple><value>ring</value><value>PBX</value><variable>callee</variable></tuple></out>
    </actions>
    <atLocations>PBX</atLocations>
  </process>
  <process id="P_PBX_answer">
    <actions>
      <in><tuple><value>call</value><value>PBX</value><input>caller</input></tuple></in>
      <out><value>IP</value><value>PBX</value><variable>caller</variable>
          <tuple><value>answer</value><value>PBX</value><variable>caller</variable></tuple></out>
    </actions>
    <atLocations>PBX</atLocations>
  </process>
  <process id="P_PBX_IPout">
    <actions>
      <in><value>IP</value><value>PBX</value><input>target</input><input>tuple</input></in>
      <out><value>IP</value><value>PBX</value><variable>target</variable>
          <variable>tuple</variable><locval>PBX</locval></out>
    </actions>
    <atLocations>PBX</atLocations>
  </process>
  <process id="P_PBX_IPin">
    <actions>
      <in><value>IP</value><input>source</input><value>PBX</value><input>tuple</input></in>
      <out><variable>tuple</variable></out>
    </actions>
    <atLocations>PBX</atLocations>
  </process>
  <process id="P_PBX_IP">
    <actions>
      <in><value>IP</value><input>source</input><input>target</input><input>tuple</input></in>
      <in><variable>target</variable><input>route</input><locval>PBX_routingtable</locval></in>
      <out><value>IP</value><variable>source</variable><variable>target</variable>
```

```
            <variable>tuple</variable><locvar>route</locvar></out>
        </actions>
        <atLocations>PBX</atLocations>
      </process>
      <process id="P_serverA_IP">
        <actions>
          <in><value>IP</value><input>source</input><input>target</input><input>tuple</input></in>
          <in><variable>target</variable><input>route</input><locval>serverA_routingtable</locval></in>
          <out><value>IP</value><variable>source</variable><variable>target</variable>
            <variable>tuple</variable><locvar>route</locvar></out>
        </actions>
        <atLocations>serverA</atLocations>
      </process>
      <process id="P_serverB_IP">
        <actions>
          <in><value>IP</value><input>source</input><input>target</input><input>tuple</input></in>
          <in><variable>target</variable><input>route</input><locval>serverB_routingtable</locval></in>
          <out><value>IP</value><variable>source</variable><variable>target</variable>
            <variable>tuple</variable><locvar>route</locvar></out>
        </actions>
        <atLocations>serverB</atLocations>
      </process>
      <process id="P_otherphone_call">
        <actions>
          <in><tuple><value>call</value><input>callee</input></tuple></in>
          <out><value>IP</value><value>phone</value><variable>callee</variable>
            <tuple><value>ring</value><value>phoneB</value><variable>callee</variable></tuple></out>
        </actions>
        <atLocations>phoneB</atLocations>
      </process>
      <process id="P_otherphone_answer">
        <actions>
          <in><tuple><value>call</value><value>phoneB</value><input>caller</input></tuple></in>
          <out><value>IP</value><value>phoneB</value><variable>caller</variable>
            <tuple><value>answer</value><value>phoneB</value><variable>caller</variable></tuple></out>
        </actions>
        <atLocations>internalPhone</atLocations>
      </process>
      <process id="P_otherphone_IPout">
        <actions>
          <in><value>IP</value><value>phoneB</value><input>target</input><input>tuple</input></in>
          <out><value>IP</value><value>phoneB</value><variable>target</variable>
            <variable>tuple</variable><locval>serverB</locval></out>
        </actions>
        <atLocations>phoneB</atLocations>
      </process>
      <process id="P_otherphone_IPin">
        <actions>
          <in><value>IP</value><input>source</input><value>phoneB</value><input>tuple</input></in>
          <out><variable>tuple</variable></out>
        </actions>
        <atLocations>phoneB</atLocations>
      </process>
    </processes>
</system>
```

B.3. Cloud Infrastructure

B.3.1. Scenario

```
<?xml version="1.0" encoding="UTF-8"?>
<scenario
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_scenario.xsd"
    id="Cloud case study"
    author="CW Probst"
    date="2016-10-18 14:23:00"
    version="1.0">
  <model>CLOUD-Model.xml</model>
  <assetGoal attacker="X">
    <asset>fileX</asset>
  </assetGoal>
</scenario>
```

B.3.2. Model

```
<?xml version="1.0" encoding="UTF-8"?>
<system
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_model"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_model.xsd"
    author="Christian W Probst"
    date="2016-10-10 11:13:25"
    version="1.0">
  <title>TREsPASS cloud infrastructure model</title>
  <locations>
    <location id="Outside"/>
    <location id="Hallway"/>
    <location id="DoorInternal"/>
    <location id="RoomInternal"/>
    <location id="WindowInternal"/>
    <location id="DoorDatacenter"/>
    <location id="RoomDatacenter"/>
    <location id="WindowDatacenter"/>
  </locations>
  <edges>
    <edge directed="false"><source>Outside</source><target>Hallway</target></edge>
    <edge><source>Hallway</source><target>DoorInternal</target></edge>
    <edge><source>DoorInternal</source><target>RoomInternal</target></edge>
    <edge><source>RoomInternal</source><target>Hallway</target></edge>
    <edge directed="false"><source>RoomInternal</source><target>WindowInternal</target></edge>
    <edge directed="false"><source>Outside</source><target>WindowInternal</target></edge>
    <edge directed="false"><source>RoomInternal</source><target>DoorDatacenter</target></edge>
    <edge directed="false"><source>DoorDatacenter</source><target>RoomDatacenter</target></edge>
    <edge directed="false"><source>WindowDatacenter</source><target>RoomDatacenter</target></edge>
    <edge directed="false"><source>Hallway</source><target>WindowDatacenter</target></edge>
    <edge directed="false"><source>Laptop</source><target>Router</target></edge>
    <edge directed="false"><source>Router</source><target>Server1</target></edge>
    <edge directed="false"><source>Router</source><target>Server2</target></edge>
    <edge directed="false"><source>Router</source><target>vm1</target></edge>
  </edges>
  <assets>
    <item name="usercard" id="usercard"><atLocations>Ethan</atLocations></item>
    <data name="userpin" id="userpin" value="1"><atLocations>Ethan usercard</atLocations></data>
    <data name="username" id="user" value="user"><atLocations>Ethan</atLocations></data>
    <data name="password" id="pwd" value="pwd"><atLocations>Ethan</atLocations></data>
    <item name="admincard" id="admincard"><atLocations>Sydney</atLocations></item>
    <data name="adminpin" id="adminpin" value="1"><atLocations>Sydney admincard</atLocations></data>
    <data name="username" id="admin" value="admin"><atLocations>Sydney</atLocations></data>
    <data name="password" id="apwd" value="apwd"><atLocations>Sydney</atLocations></data>
    <item name="Laptop" id="Laptop"><atLocations>RoomInternal</atLocations></item>
    <item name="Router" id="Router"><atLocations>RoomDatacenter</atLocations></item>
    <item name="routingtable" id="routingtable"><atLocations>Router</atLocations></item>
    <data id="route1" name="Laptop" value="Laptop"><atLocations>routingtable</atLocations></data>
    <data id="route2" name="Server1" value="Server1"><atLocations>routingtable</atLocations></data>
    <data id="route3" name="Server2" value="Server2"><atLocations>routingtable</atLocations></data>
    <data id="route4" name="vm1" value="vm1"><atLocations>routingtable</atLocations></data>
    <item name="Server1" id="Server1"><atLocations>RoomDatacenter</atLocations></item>
    <item name="Server2" id="Server2"><atLocations>RoomDatacenter</atLocations></item>
    <item name="vm1" id="vm1"><atLocations>Server1</atLocations></item>
    <data name="fileX" id="fileX" value="42"><atLocations>vm1</atLocations></data>
  </assets>
  <actors>
    <actor id="Ethan"><atLocations>Outside</atLocations></actor>
    <actor id="Sydney"><atLocations>Outside</atLocations></actor>
    <actor id="Cleo"><atLocations>Outside</atLocations></actor>
```

```
</actors>
<policies>
    <policy id="p001">
        <credentials>
            <credItem name="usercard"><credData name="userpin"><variable>Y</variable></credData></credItem>
            <credData name="userpin"><variable>Y</variable></credData>
        </credentials>
        <enabled><move /></enabled>
        <atLocations>DoorInternal</atLocations>
    </policy>
    <policy id="p002">
        <credentials>
            <credItem name="admincard"><credData name="adminpin"><variable>Y</variable></credData></credItem>
            <credData name="adminpin"><variable>Y</variable></credData>
        </credentials>
        <enabled><move /></enabled>
        <atLocations>DoorInternal</atLocations>
    </policy>
    <policy id="p003">
      <credentials>
        <credData name="password"><value>pwd</value></credData>
        <credData name="username"><value>user</value></credData>
      </credentials>
      <enabled><out><tuple><value>getfile</value><wildcard/><wildcard/></tuple></out>
      </enabled>
      <atLocations>Laptop</atLocations>
    </policy>
    <policy id="p004">
      <credentials>
        <credData name="password"><value>apwd</value></credData>
        <credData name="username"><value>admin</value></credData>
      </credentials>
      <enabled><out><tuple><value>move</value><wildcard/><wildcard/></tuple></out>
      </enabled>
      <atLocations>Server1 Server2</atLocations>
    </policy>
    <policy id="p005">
      <credentials></credentials>
      <enabled></enabled>
      <atLocations>WindowInternal WindowDatacenter</atLocations>
    </policy>
</policies>
<processes>
  <process id="P_query">
    <actions>
      <in><tuple><value>getfile</value><input>Server</input><input>filename</input></tuple></in>
      <out><value>IP</value><value>Laptop</value><variable>Server</variable>
          <tuple><value>get</value><value>Laptop</value><variable>filename</variable></tuple></out>
      <in><tuple><value>file</value><variable>filename</variable><input>content</input></tuple></in>
      <out><tuple><variable>filename</variable><variable>content</variable></tuple></out>
    </actions>
    <atLocations>Laptop</atLocations>
  </process>
  <process id="P_IPin_Laptop">
    <actions>
      <in><value>IP</value><input>source</input><value>Laptop</value><input>tuple</input></in>
      <out><variable>tuple</variable></out>
    </actions>
    <atLocations>Laptop</atLocations>
  </process>
  <process id="P_IPout_Laptop">
    <actions>
      <in><value>IP</value><value>Laptop</value><input>target</input><input>tuple</input></in>
      <out><value>IP</value><value>Laptop</value><variable>target</variable>
          <variable>tuple</variable><locval>Router</locval></out>
    </actions>
    <atLocations>Laptop</atLocations>
  </process>
<process id="P_IPin_Server1">
  <actions>
    <in><value>IP</value><input>source</input><value>Server1</value><input>tuple</input></in>
    <out><variable>tuple</variable></out>
  </actions>
  <atLocations>Server1</atLocations>
</process>
<process id="P_IPout_Server1">
  <actions>
    <in><value>IP</value><value>Server1</value><input>target</input><input>tuple</input></in>
    <out><value>IP</value><value>Server1</value><variable>target</variable>
        <variable>tuple</variable><locval>Router</locval></out>
  </actions>
  <atLocations>Server1</atLocations>
</process>
<process id="P_IPin_Server2">
  <actions>
    <in><value>IP</value><input>source</input><value>Server2</value><input>tuple</input></in>
    <out><variable>tuple</variable></out>
  </actions>
```

```
      <atLocations>Server2</atLocations>
    </process>
    <process id="P_IPout_Server2">
      <actions>
        <in><value>IP</value><value>Server2</value><input>target</input><input>tuple</input></in>
        <out><value>IP</value><value>Server2</value><variable>target</variable>
            <variable>tuple</variable><locval>Router</locval></out>
      </actions>
      <atLocations>Server2</atLocations>
    </process>
    <process id="P_IPin_vm1">
      <actions>
        <in><value>IP</value><input>source</input><value>vm1</value><input>tuple</input></in>
        <out><variable>tuple</variable></out>
      </actions>
      <atLocations>vm1</atLocations>
    </process>
    <process id="P_IPout_vm1">
      <actions>
        <in><value>IP</value><value>vm1</value><input>target</input><input>tuple</input></in>
        <out><value>IP</value><value>vm1</value><variable>target</variable>
            <variable>tuple</variable><locval>Router</locval></out>
      </actions>
      <atLocations>vm1</atLocations>
    </process>
    <process id="P_IPRouter">
      <actions>
        <in><value>IP</value><input>source</input><input>target</input><input>tuple</input></in>
        <in><variable>target</variable><input>route</input><locval>routingtable</locval></in>
        <out><value>IP</value><variable>source</variable><variable>target</variable>
            <variable>tuple</variable><locvar>route</locvar></out>
      </actions>
      <atLocations>Router</atLocations>
    </process>
    <process id="P_vm1_ftp">
      <actions>
        <in><tuple><value>get</value><input>sender</input><input>filename</input></tuple></in>
        <in><variable>filename</variable><input>value</input></in>
        <out><value>IP</value><value>vm1</value><variable>sender</variable>
            <tuple><value>file</value><variable>filename</variable><variable>value</variable></tuple></out>
      </actions>
      <atLocations>vm1</atLocations>
    </process>
    <process id="P_server1_move">
      <actions>
        <in><tuple><value>move</value><input>server</input><input>vmname</input></tuple></in>
        <in><variable>vmname</variable><input>content</input></in>
        <out><value>IP</value><value>Server1</value><variable>server</variable>
            <tuple><value>vm</value><variable>vmname</variable><variable>content</variable></tuple></out>
      </actions>
      <atLocations>Server1</atLocations>
    </process>
    <process id="P_server1_receive">
      <actions>
        <in><tuple><value>vm</value><input>vmname</input><input>content</input></tuple></in>
        <out><variable>vmname</variable><variable>content</variable></out>
      </actions>
      <atLocations>Server1</atLocations>
    </process>
    <process id="P_server2_move">
      <actions>
        <in><tuple><value>move</value><input>server</input><input>vmname</input></tuple></in>
        <in><variable>vmname</variable><input>content</input></in>
        <out><value>IP</value><value>Server2</value><variable>server</variable>
            <tuple><value>vm</value><variable>vmname</variable><variable>content</variable></tuple></out>
      </actions>
      <atLocations>Server2</atLocations>
    </process>
    <process id="P_server2_receive">
      <actions>
        <in><tuple><value>vm</value><input>vmname</input><input>content</input></tuple></in>
        <out><variable>vmname</variable><variable>content</variable></out>
      </actions>
      <atLocations>Server2</atLocations>
    </process>
    </processes>
</system>
```

B.4. ATM

B.4.1. Scenario

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<scenario
    xmlns="https://www.trespass-project.eu/schemas/TREsPASS_scenario"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_scenario
                        https://www.trespass-project.eu/schemas/TREsPASS_scenario.xsd"
    author="Christian W Probst"
    version="1.0"
    date="2016-10-26 17:21:00"
    id="ATM_scenario">
  <assetGoal attacker="actor__attacker" profit="5000">
    <asset>Money</asset>
  </assetGoal>
  <model>ATM-Model.xml</model>
</scenario>
```

B.4.2. Model

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<system
  xmlns="https://www.trespass-project.eu/schemas/TREsPASS_model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://www.trespass-project.eu/schemas/TREsPASS_model
                      https://www.trespass-project.eu/schemas/TREsPASS_model.xsd"
  author="Christian W Probst"
  version="1.0.0"
  date="2016-10-10 09:01:28"
  id="ATM Model">
  <title>ATM Model</title>
  <actors>
    <actor id="Cashier" name="Cashier"><atLocations>LisbonCity</atLocations></actor>
    <actor id="GasStationIntruder" name="GasStationIntruder"><atLocations>GasStation</atLocations></actor>
    <actor id="Customer" name="Customer"><atLocations>NeighborhoodA</atLocations></actor>
    <actor id="ATMMaintenance" name="ATMMaintenance"><atLocations>ATMNetworkProvider</atLocations></actor>
    <actor id="ATMAdministrator" name="ATMAdministrator"><atLocations>Bank</atLocations></actor>
    <actor id="Attacker" name="Attacker"><atLocations>LisbonCity</atLocations></actor>
    <actor id="BankIntruder" name="BankIntruder"><atLocations>Bank</atLocations></actor>
  </actors>
  <locations>
    <location id="GasStation" name="GasStation"/>
    <location id="NeighborhoodA" name="NeighborhoodA"/>
    <location id="LisbonCity" name="LisbonCity"/>
    <location id="Window" name="Window"/>
    <location id="Door" name="Door"/>
    <location id="Bank" name="Bank"/>
    <location id="OtherBanks" name="OtherBanks"/>
    <location id="ATMNetworkProvider" name="ATMNetworkProvider"/>
  </locations>
  <assets>
    <item id="card" name="card"><atLocations>Customer</atLocations></item>
    <data id="pin" name="pin" value="42"><atLocations>card Customer</atLocations></data>
    <item id="ExternalCamera" name="ExternalCamera"><atLocations>GasStation</atLocations></item>
    <item id="InternalCamera" name="InternalCamera"><atLocations>GasStation</atLocations></item>
    <item id="HiddenCamera" name="HiddenCamera"><atLocations>GasStation</atLocations></item>
    <item id="DoorSensor" name="DoorSensor"><atLocations>Door</atLocations></item>
    <item id="AntiRamRaidBars" name="AntiRamRaidBars"><atLocations>GasStation</atLocations></item>
    <item id="ATM" name="ATM"><atLocations>GasStation</atLocations></item>
    <item id="SafeBox" name="SafeBox"><atLocations>ATM</atLocations></item>
    <item id="CashCassettes" name="CashCassettes"><atLocations>SafeBox</atLocations></item>
    <item id="Money" name="Money"><atLocations>CashCassettes</atLocations></item>
    <item id="Display" name="Display"><atLocations>ATM</atLocations></item>
    <item id="Printer" name="Printer"><atLocations>ATM</atLocations></item>
    <item id="EPP" name="EPP"><atLocations>ATM</atLocations></item>
    <item id="CardReader" name="CardReader"><atLocations>ATM</atLocations></item>
    <item id="CashDispenser" name="CashDispenser"><atLocations>ATM</atLocations></item>
    <item id="ATMKey" name="ATMKey"><atLocations>ATMMaintenance</atLocations></item>
    <item id="SafeBoxKey" name="SafeBoxKey"><atLocations>ATMMaintenance</atLocations></item>
    <item id="WindowsSensor" name="WindowsSensor"><atLocations>Window</atLocations></item>
    <item id="GasStationKey" name="GasStationKey"><atLocations>Cashier</atLocations></item>
    <item id="BurglarAlarm" name="BurglarAlarm"><atLocations>GasStation</atLocations></item>
    <item id="BankSwitch" name="BankSwitch"><atLocations>Bank</atLocations></item>
    <item id="ATMNetwork" name="ATMNetwork"><atLocations>ATMNetworkProvider</atLocations></item>
    <item id="BankCoreServer" name="BankCoreServer"><atLocations>Bank</atLocations></item>
    <item id="ATMPC" name="ATMPC"><atLocations>ATM</atLocations></item>
  </assets>
  <edges>
    <edge directed="false"><source>Bankswitch</source><target>ATMNetwork</target></edge>
```

```
    <edge directed="false"><source>ATMPC</source><target>Bankswitch</target></edge>
    <edge directed="false"><source>Bankswitch</source><target>BankCoreServer</target></edge>
    <edge directed="false"><source>ATMPC</source><target>CashCassettes</target></edge>
    <edge directed="false"><source>OtherBanks</source><target>ATMNetwork</target></edge>
    <edge directed="false"><source>DoorSensor</source><target>BurglarAlarm</target></edge>
    <edge directed="false"><source>WindowsSensor</source><target>BurglarAlarm</target></edge>
    <edge directed="false"><source>GasStation</source><target>Window</target></edge>
    <edge directed="false"><source>GasStation</source><target>Door</target></edge>
    <edge directed="false"><source>Door</source><target>NeighborhoodA</target></edge>
    <edge directed="false"><source>NeighborhoodA</source><target>LisbonCity</target></edge>
    <edge directed="false"><source>Bank</source><target>LisbonCity</target></edge>
    <edge directed="false"><source>OtherBanks</source><target>LisbonCity</target></edge>
    <edge directed="false"><source>ATMNetworkProvider</source><target>LisbonCity</target></edge>
    <edge directed="false"><source>Window</source><target>NeighborhoodA</target></edge>
  </edges>
  <policies>
    <policy id="pol001">
      <credentials><credItem name="GasStationKey"/></credentials>
      <enabled><move/></enabled>
      <atLocations>Door</atLocations>
    </policy>
    <policy id="pol002">
      <credentials><credItem name="SafeBoxKey"/></credentials>
      <enabled><in/></enabled>
      <atLocations>safe-box</atLocations>
    </policy>
    <policy id="pol003">
      <credentials><credItem name="ATMKey"/></credentials>
      <enabled><in/></enabled>
      <atLocations>ATM</atLocations>
    </policy>
  </policies>
  <processes>
    <process id="process__ATM"><atLocations>ATM</atLocations>
      <actions>
        <in><value>card</value><input>card</input><locval>CardReader</locval></in>
        <in><value>pin</value><input>userpin</input><locval>EPP</locval></in>
        <in><value>pin</value><variable>userpin</variable><locvar>card</locvar></in>
        <in><value>account</value><input>account</input><locvar>card</locvar></in>
        <in><tuple><value>withdraw</value><input>amount</input></tuple><locval>EPP</locval></in>
        <out><value>IP</value><value>ATM</value><value>BankCoreServer</value>
            <tuple><value>withdraw</value><variable>account</variable><variable>amount</variable><value>ATM</value></tuple></out>
        <in><tuple><value>withdraw</value><value>ok</value></tuple></in>
        <in><value>Money</value><input>cash</input><locval>CashCassettes</locval></in>
        <out><variable>card</variable><locval>CardReader</locval></out>
        <out><tuple><value>Money</value><variable>cash</variable></tuple><locval>CashDispenser</locval></out>
      </actions>
    </process>
    <process id="P_IPin_ATM"><atLocations>ATM</atLocations>
      <actions>
        <in><value>IP</value><input>source</input><value>ATM</value><input>tuple</input></in>
        <out><variable>tuple</variable></out>
      </actions>
    </process>
    <process id="P_IPout_ATM"><atLocations>ATM</atLocations>
      <actions>
        <in><value>IP</value><value>ATM</value><input>target</input><input>tuple</input></in>
        <out><value>IP</value><value>item</value><variable>target</variable>
            <variable>tuple</variable><locval>BankCoreServer</locval></out>
      </actions>
    </process>
    <process id="P_IPin_BCS"><atLocations>BankCoreServer</atLocations>
      <actions>
        <in><value>IP</value><input>source</input><value>BankCoreServer</value><input>tuple</input></in>
        <out><variable>tuple</variable></out>
      </actions>
    </process>
    <process id="P_IPout_BCS"><atLocations>BankCoreServer</atLocations>
      <actions>
        <in><value>IP</value><value>BankCoreServer</value><input>target</input><input>tuple</input></in>
        <out><value>IP</value><value>BankCoreServer</value><variable>target</variable>
            <variable>tuple</variable><locval>ATM</locval></out>
      </actions>
    </process>
    <process id="P_BCS"><atLocations>BankCoreServer</atLocations>
      <actions>
        <in><tuple><value>withdraw</value><input>account</input><input>amount</input><input>sender</input></tuple></in>
        <in><value>Money</value><variable>amount</variable><locvar>account</locvar></in>
        <out><value>IP</value><value>BankCoreServer</value><variable>sender</variable>
            <tuple><value>withdraw</value><value>ok</value></tuple></out>
      </actions>
    </process>
  </processes>
</system>
```