Pre-proceedings of

The First International Workshop on Graphical Models for Security

GraMSec'14

Preface

The present volume contains the pre-proceedings of *The First International Workshop on Graphical Models for Security* (GraMSec'14). The workshop was held in Grenoble, France, on April 12, 2014, as one of the satellite events of *The European Joint Conferences on Theory and Practice of Software* (ETAPS) 2014.

Graphical security models provide an intuitive but systematic methodology to analyze security weaknesses of systems and to evaluate potential protection measures. Such models have been subject of academic research and they have also been widely accepted by the industrial sector, as a means to support and facilitate threat analysis and risk management processes.

The objective of the *International Workshop on Graphical Models for Security* is to contribute to the development of well-founded graphical security models, efficient algorithms for their analysis, as well as methodologies for their practical usage. The workshop brings together academic researchers and industry practitioners designing and employing visual models for security in order to provide a platform for discussion, knowledge exchange and collaborations.

Thirteen submissions were received by this first edition of GraMSec and each of them was reviewed by at least three reviewers. Based on their quality and contribution to the field, six papers, presented in this volume, were accepted for presentation at the workshop and inclusion in the final proceedings of GraMSec'14.

We would like to thank all the authors for submitting their work to GraMSec'14 and the members of the Program Committee as well as external reviewers for their efforts and high-quality reviews. We are also grateful to the organizers of ETAPS 2014, especially to the Workshops Chair Axel Legay, for accepting GraMSec'14 as an ETAPS-affiliated event and for providing a perfect environment for running the workshop. We would like to thank the *Fonds National de la Recherche Luxembourg* and the *European Commission's Seventh Framework Programme* for their partial sponsorship of the workshop (FNR-CORE ADT2P grant and the EU FP7 grant no. 318003 TREsPASS). Finally, we are thankful to the University of Luxembourg, the University of Twente, and Delft University of Technology for their in kind contribution to GraMSec'14.

February 2014

Sjouke Mauw
Barbara Kordy
Wolter Pieters

GraMSec'14 Organizing Committees

General Chair

Prof. Dr. Sjouke Mauw, University of Luxembourg, Luxembourg

Program Co-chairs

Dr. Barbara Kordy, University of Luxembourg, Luxembourg
Dr. Wolter Pieters, Delft University of Technology and University of Twente, The Netherlands

Program Committee

Giampaolo Bella, University of Catania, Italy
Matt Bishop, University of California at Davis, USA
Stefano Bistarelli, University of Perugia, Italy
Mathias Ekstedt, KTH Royal Institute of Technology, Sweden
Donald Firesmith, Software Engineering Institute, USA
Virginia N. L. Franqueira, University of Central Lancashire, UK
Paolo Giorgini, University of Trento, Italy
Siv Hilde Houmb, Secure-NOK AS and Gjvik University College, Norway
Sushil Jajodia, George Mason University, USA
Henk Jonkers, BiZZdesign, The Netherlands
Jan Jürjens, Technical University Dortmund, Germany
Peter Karpati, Institute for Energy Technology, Norway
Dong Seong Kim, University of Canterbury, New Zealand
Gabriele Lenzini, University of Luxembourg, Luxembourg
Per Håkon Meland, SINTEF, Norway
Svetla Nikova, KU Leuven, Belgium
Andreas L. Opdahl, University of Bergen, Norway
Stéphane Paul, Thales Research and Technology, France
Milan Petkovic, Philips and Eindhoven University of Technology, The Netherlands
Ludovic Piètre-Cambacédès, EDF, France
Christian W. Probst, Technical University of Denmark, Denmark
William H. Sanders, University of Illinois, USA
Simone Sillem, Delft University of Technology, The Netherlands
Guttorm Sindre, Norwegian University of Science and Technology, Norway
Mariëlle Stoelinga, University of Twente, The Netherlands
Kishor S. Trivedi, Duke University, USA
Luca Viganò, King's College London, UK
Lingyu Wang, Concordia University, Canada
Jan Willemson, Cybernetica, Estonia

External Reviewers

Elisa Costante, Eindhoven University of Technology, The Netherlands
Dennis Guck, University of Twente, The Netherlands
Hugo Jonker, University of Luxembourg, Luxembourg
Ali Koudri, Thales Research and Technology, France
Zhan Wang, George Mason University, USA

Papers Accepted to GraMSec'14

- Erlend Andreas Gjære and Per Håkon Meland
  Threats Management Throughout the Software Service Life-Cycle
- Ludovic Apvrille and Yves Roudier
  Towards the Model-Driven Engineering of Secure yet Safe Embedded Systems
- Stéphane Paul
  Towards Automating the Construction & Maintenance of Attack Trees: a Feasibility Study
- Thomas Bauereiss and Dieter Hutter
  Possibilistic Information Flow Control for Workflow Management Systems
- Cristian Prisacariu
  Actor Network Procedures as Psi-calculi for Security Ceremonies
- Aitor Couce Vieira, Siv Hilde Houmb and David Rios Insua
  A Graphical Adversarial Risk Analysis Model for Oil and Gas Drilling Cybersecurity

Threats management throughout the software service life-cycle

Erlend Andreas Gjære
SINTEF ICT
Trondheim, Norway
erlendandreas.gjare@sintef.no

Per Håkon Meland
SINTEF ICT
Trondheim, Norway
per.h.meland@sintef.no

Software services are inevitably exposed to a fluctuating threat picture. Unfortunately, not all threats can be handled only with preventive measures during design and development, but also require adaptive mitigations at runtime. In this paper we describe an approach where we model composite services and threats together, which allows us to create preventive measures at design-time. At runtime, our specification also allows the service runtime environment (SRE) to receive alerts about active threats that we have not handled, and react to these automatically through adaptation of the composite service. A goal-oriented security requirements modelling tool is used to model business-level threats and analyse how they may impact goals. A process flow modelling tool, utilising Business Process Model and Notation (BPMN) and standard error boundary events, allows us to define how threats should be responded to during service execution on a technical level. Throughout the software life-cycle, we maintain threats in a centralised threat repository. Re-use of these threats extends further into monitoring alerts being distributed through a cloud-based messaging service. To demonstrate our approach in practice, we have developed a proof-of-concept service for the Air Traffic Management (ATM) domain. In addition to the design-time activities, we show how this composite service duly adapts itself when a service component is exposed to a threat at runtime.

1 Introduction

With the emerging paradigm of composite services, i.e. software services which are composed of functionality provided by several services components, possibly involving several service providers, an attack on a service component implies an attack on the composite service as a whole. Service compositions in general are highly distributed and have a complex nature, exposing a greater attack surface than traditional stand-alone systems. To make this kind of services secure enough, it is vital to take a holistic approach to how they are built, security-wise. This assumes incorporation of activities that deal with threats at many stages throughout the life-cycle, comprising both design and development as well as the runtime phase.

At design-time, we can utilise preventive activities for mitigating threats, and at runtime there should be performed corrective activities to handle the residual threats. It is very optimistic to think that all threats are known and correctly assessed from the very beginning, e.g. it is not given that we have reliable knowledge of motivation and available resources for a potential attacker or enough resources to implement preventive measures. What we do know, is that when a threat escalates, we need to make sure that our organisation and the system(s) we are trying to protect are prepared to respond effectively.

Our main goal with this paper is to describe a process and tool-chain which combines threat modelling with graphical goal and process models to derive preventive measures, such as security requirements for development, and also enables further corrective measures to be automatically applied at runtime. We have chosen to support security analysis through activities and tools that are already adopted and in use, although these have not necessarily been previously combined.

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

The details of our proposed approach is given in section 2, incorporating our Air Traffic Management system case study. Through this we explain and demonstrate the use of a threat repository to facilitate sharing and re-use of threats, goal-oriented modelling with threats, service design with BPMN, model transformation, threat response recommendation, defining rules for dealing with escalations, and finally runtime management of composite services. Section 3 discusses strengths and compromises made in this approach, and section 4 concludes the paper.

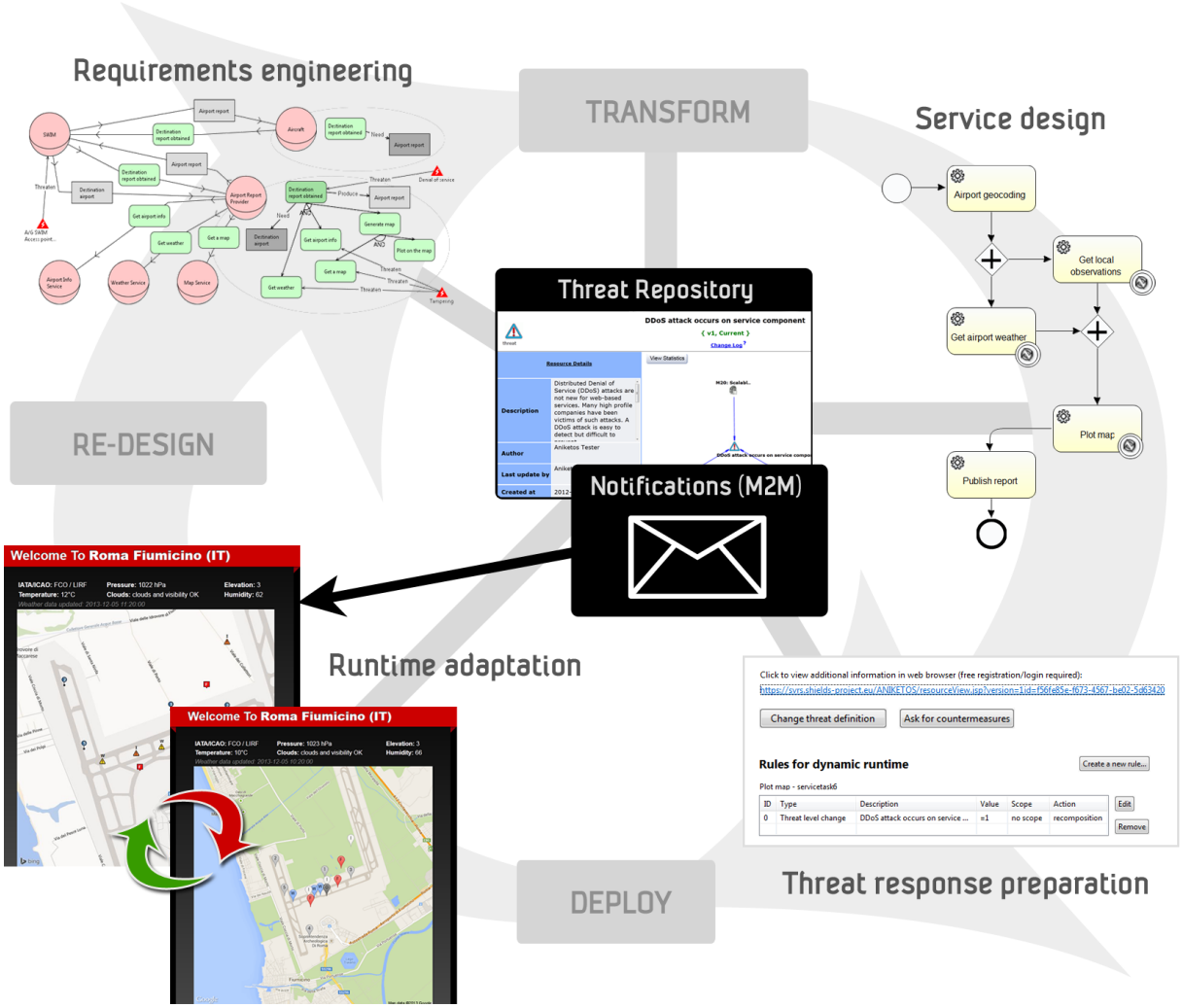

Figure 1: A software life-cycle model showing how threats can be modelled and managed at design-time and prepared for automated service adaptation at runtime.

2 Approach description

Figure 1 gives a visual overview of the threat management life-cycle, outlining both activities in the process as well as data flow. Key to understanding the advantages of this holistic view are the threat repository and notification services in the centre, which together connect the dots between the remaining

components and stages. In the requirements engineering process we utilise goal-oriented modelling (Socio-Technical Security modelling language, STS-ml [18]) to elicit and define security requirements. Here we also instantiate some of the main threats to our system on an organisational/business level of abstraction. Further, we use Business Process Model and Notation (BPMN) [13] for designing the service process of a composite web service. At this stage we do not depend upon how the requirements engineering has been done, but if STS-ml has been used, we can transform the goal models into process models with some help from a software tool. The BPMN model defines a service process which can be deployed and executed as a web service, and here should the threats be refined on a more technically detailed level. Based on this, we can ask for accordingly technical countermeasures from the threat repository. We can also define rules for responding to particular events, if we are able to monitor the service components at runtime. Whenever an alert is received from the monitors, these rules will be evaluated by a service runtime environment (SRE) tailored for this purpose. If is a match is found, a re-composition can for example be triggered, replacing a service component instance with another one providing similar functionality. Alternatively, one can move along in the life-cycle to re-design of the system and/or the particular service.

2.1 Case study: Air Traffic Management

For demonstrating our proof-of-concept, we have chosen a case study based on Air Traffic Management (ATM). The European air navigation services and their supporting systems are currently undergoing a paradigm shift, most notably through the System Wide Information Management (SWIM) [6]. Going from numerous incompatible stand-alone systems, SWIM enables cross-border collaboration and data exchange between many systems and organisations with its service oriented architecture (SOA) approach. This raises prospects for an expansive registry of composite services, which hopefully will contribute to maximizing efficiency of the airspace with time. The service we demonstrate is one that gathers various information about an airport to be used e.g. in the cases when a pilot wants a fresh report on the flight's destination. As seen from the perspective of this service provider, further details on this composite service sample is provided along with the description of each step in our approach below. Security-experts from three ATM organisations were involved in the modelling, using already defined security requirements and system specifications from the SESAR project [17] through which SWIM is developed. However, the models provided in this paper only represent a limited excerpt of the SWIM system model, and the demonstrator we have implemented is not based on service components actually provided through SWIM. Work on collecting and analysing feedback from the actual case study users is yet to be completed, and results as such are hence not considered in this paper.

2.2 Knowledge management through a threat repository

Every part of our approach is tied together by the concept of having a centralised repository of threats [9]. While threats can always be created and modelled independently in the different diagram types that we use, a persistent threat repository provides better conditions for import and re-use of threats [12]. It is of essence here that each threat in the threat repository is associated with a unique threat identifier (ID), and that this threat ID is stored along with the threats in the various diagrams we create. This allows the tools to always access (meta-) information on threats stored in the repository, and for the threats to be passed between the models and pushed further through deployment into the runtime phase.

In our proof-of-concept, we have utilised an existing online threat repository service [16], providing an application programming interface (API) for accessing its large collection threats externally. The API

offers functionality for searching for threats in terms of their name, class (business-level or operational level), or business domain tags. Some additional meta-data may also be available, such as a textual description and links to further information resources. For the two graphical modelling tools we present below, we have implemented a plug-in that takes advantage of this API, as shown in Figure 2. In addition to the API, we have access to a web-based user interface which allows more info to be looked up on each threat, including suggestions for countermeasures and relationships with other threats. A threat uploader tool has also been developed for adding new threats to the threat repository, using existing diagrams.

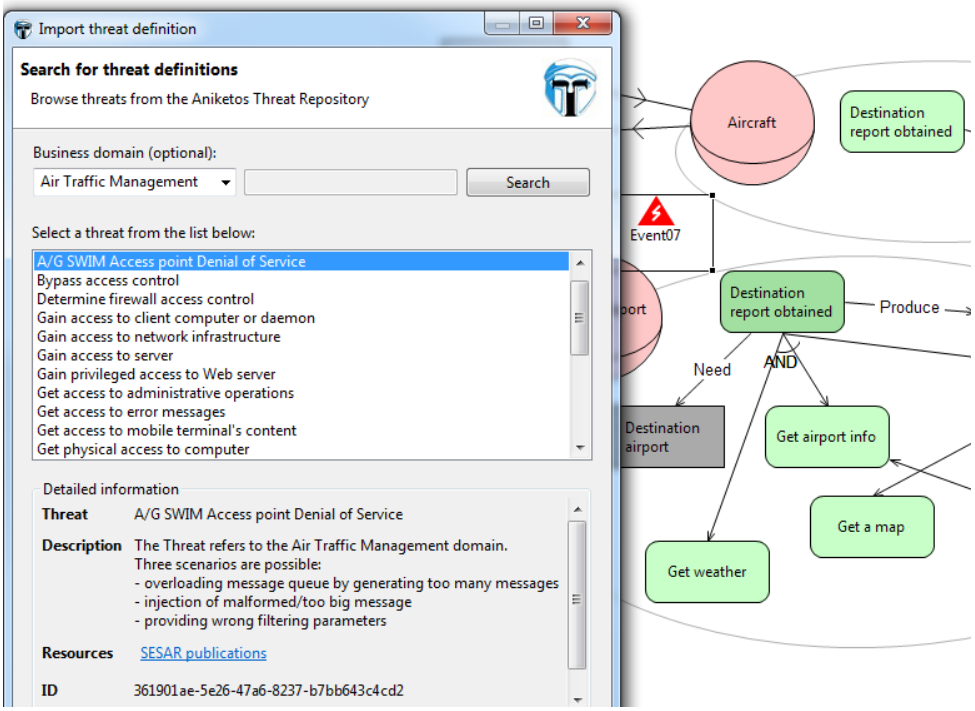

Figure 2: Importing a domain-specific threat from the threat repository into a graphical model.

Threats related specifically to ATM are found in the threat repository under the business domain selector *Air Traffic Management*. These threats can both be general ones that are known to also apply in that particular domain, such as *Gain access to server*, as well as threats more specific to ATM only – e.g. *A/G SWIM Access Point Denial of Service*. Generic high-level threats, like *Tampering*, can for instance be specialised into a domain specific threat such as *False airport coordinates*.

2.3 Requirements engineering

While requirements engineering can take many shapes and forms, graphical goal-oriented modelling is an approach well suited for complex and distributed socio-technical systems (of systems) [14, 15]. Where goals justify why the system and its functions are needed, threats justify why *security* for the system is needed.

In the STS model from our case study, depicted in Figure 3, we see that the pilot's request for a destination report is transmitted to SWIM rather than directly to a specific service provider. Since the requesting aircraft's destination is information already known inside SWIM, the request is complemented with this information here. Through its service registry, SWIM can select and invoke an actual service

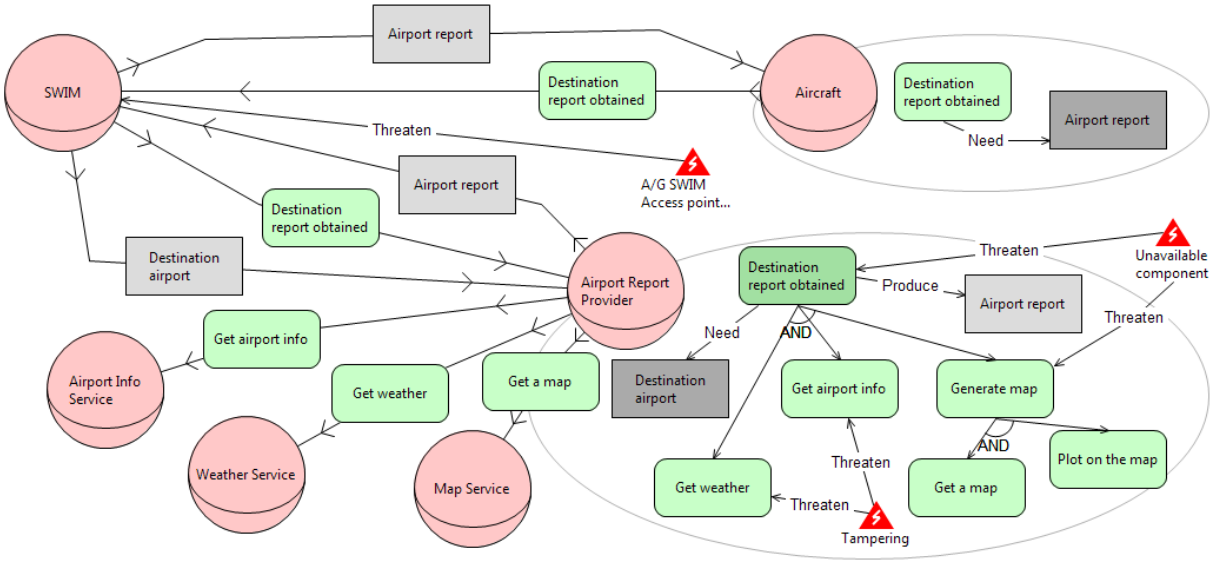

Figure 3: Goal-oriented model (STS-ml) of the airport report service – the red triangles are threats.

provider on behalf of the pilot/aircraft, possibly depending on the particular destination's location. The airport report contains various local information on the airport, along with weather conditions and a map onto which various local observations (e.g. wind meter readings, contaminations) are plotted. Threats from the perspective of the airport report provider, would in this case be any kind of tampering with externally acquired data or unavailability of any of the service components, which might hold them accountable for impacting flight safety. A threat worth including could also be denial of service within the SWIM area of responsibility, since that could block an airport report to arrive timely on the provider's behalf – although not being the airport report service provider's actual responsibility in the end.

Supported by the STS-Tool [18], we are able to both perform both graphical modelling as well as advanced formal analyses of the models. This includes checking consistency and detecting conflicts between goals and requirements, as well as analysing (visually in the model) how threats can propagate throughout the modelled system [11]. The STS-Tool also supports generating a security requirements document which contains the information we have added on threats. In addition, the document generator performs the aforementioned threat propagation analysis so that the results can be output in textual form, as shown in Figure 4. In our case specifically, the *A/G SWIM Access Point Denial of Service* threat does not have a propagated impact, and is hence not mentioned in the document, as it is directed at an actor external to the modelled system owner. Although not implemented, the document generator could potentially use the threat IDs provided in the model to look up information on countermeasures, and hence include that in the document for further reference.

2.4 Service design

In the BPMN model, we specify the process flow which our composite service shall follow. Herein lies the advantage of standardised BPMN as the service process language, that the graphical models translate into well-defined execution semantics, which in turn can be executed in business process model engines as a web service. Although BPMN has no explicit language construct for threats, we have in previous work concluded that e.g. the standard *ErrorBoundaryEvent* element can be used for representing threats

Text	Description
Impact of event Tampering in the diagram	The event Tampering threatening Get airport info and Get weather, threatens also Airport report, Destination report obtained, Airport report, Destination report obtained, Airport report and Destination report obtained.
Impact of event Unavailable component in the diagram	The event Unavailable component threatening Destination report obtained and Generate map, threatens also Airport report, Airport report, Plot on the map, Destination report obtained, Airport report, Get airport info, Get weather, Destination report obtained and Get a map.

Figure 4: Propagation of threats can be analysed with the generated security requirements document.

[10].

While our BPMN model in Figure 5 may appear similar to the STS model, the service tasks (yellow boxes) now represent atomic components of a process, unambiguously ordered as an executable process flow. Moreover, each service task implies a single invocation of a particular web service, taking process variables as (optional) input to operations defined by each service components' web service definition file (Web Service Definition Language, WSDL [20]). Although not visible in the model, the first service task, *Airport geocoding*, takes as input the IATA code of an airport (e.g. *FCO* for Rome, Fiumicino) and queries a public airport information service. The returned value, which is also defined by the component's WSDL file, should be a pair of coordinates which pinpoint the airport's geographical location. These coordinates are in turn used as input for the following two tasks, which can be done in parallel, namely to obtain the weather and any local observations from/surrounding the location in question. A fourth service takes care of plotting the gathered information on a map, whereas the final (internally maintained) service wraps up and creates the report data to be returned to the pilot's application.

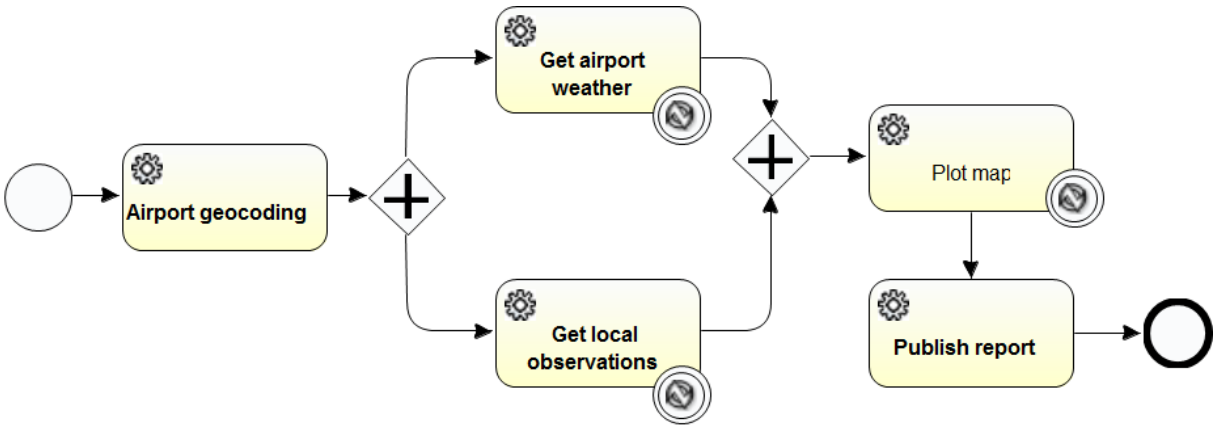

Figure 5: Service process specified in BPMN, using standard ErrorBoundaryEvent elements for threats.

We have extended the Activiti Designer tool [1], built as a plug-in for the Eclipse IDE [8], in order to support our BPMN modelling. We have chosen to store the threat ID in the BPMN 2.0 XML file, as shown in Figure 6, which is in line with the standard. This results in compatibility between different modelling tools, at least when it comes to portability of the graphical model. What is not part of BPMN is functionality for producing several alternative composition plans, i.e. alternative combinations of service components, that each provide the very same functionality [21]. This is possible when you have

```
<boundaryEvent id="boundaryerror11" attachedToRef="servicetask6"
    name="DDoS attack occurs on service component">

  <errorEventDefinition
      errorRef="f56fe85e-f673-4567-be02-5d63420b4fb4">
  </errorEventDefinition>
</boundaryEvent>
```

Figure 6: Using an ErrorEventDefinition XML-element with errorRef attribute to store the threat ID.

more than one possible candidate services for a service task, while still maintaining the required level of functionality. In our proof-of-concept, we have two candidates for the map service. The total number of composition plans equals the Cartesian product of all service component candidates for all service tasks, i.e. all possible combinations of service implementations. These different composition plans can in turn be ranked against one or several criteria, such as level of trustworthiness, quality-of-service, etc., if this information is maintained on behalf of the components [4, 5].

2.4.1 Model transformation

In order to support security information from the requirements engineering phase to be maintained at development time, essential parts from the goal models are possible to transform into a simple process model skeleton. The STS-Tool is not only, as previously described, able to output a textual description of the security requirements and threats. The STS-models can also be output to a machine-readable security requirements specification (SRS) format for use in the transformation. Although there is no way to completely automate the transformation from STS to BPMN, a transformation tool can provide a structured way to manually support selection of threats that are relevant for the service process model [2], as shown in Figure 7.

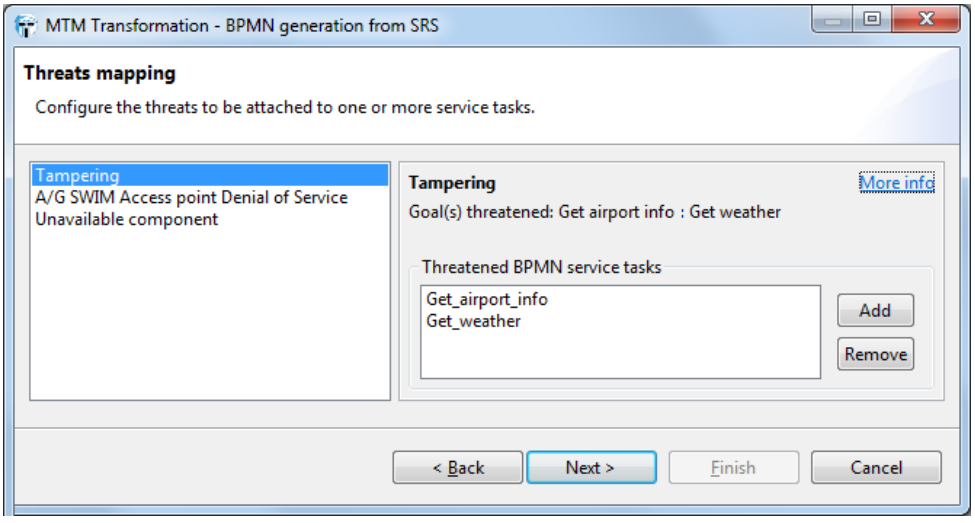

Figure 7: Tool-supported model transformation of threats from STS-ml (via SRS) to BPMN.

Transformation of STS-ml threats into BPMN threats is preferable in cases where we have input from monitors that can give the modelled threats a meaningful role at runtime. An advantage of the tool-

supported model transformation is that the threat ID, which enables linking a modelled threat to the threat repository, can be conveniently transferred between the models. In that way, any further transformation and use of the threat can be done with support from the threat repository, and the information always to be found here.

In our case study, we have *Unavailable component* as a threat in the goal model, which can be specialised into e.g. a *DDoS attack on service component* threat in the service process. This threat can be fairly easy to monitor, and we hence are potentially able adapt our service in case this threat escalates.

2.4.2 Threat response recommendations

A separate component, also integrated with the online threat repository, implements logic and knowledge to find and recommend possible mitigations for threats [3]. These countermeasures can potentially be provided in various formats, ranging from textual descriptions to actual code or services.

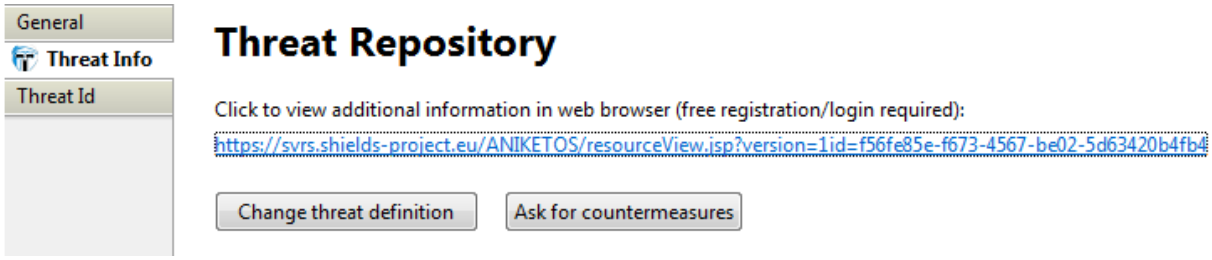

Figure 8: Threat ID is maintained so the service process modelling tool can ask for countermeasures.

In order to obtain the threat response recommendations, the threat IDs are gathered from the BPMN diagram and the relevant *ErrorBoundaryEvent* elements. Again, the threat ID is input to the threat repository, and returned are the countermeasures. As there may be several countermeasures available for each threat, the various options are ranked before being presented to the service designer.

2.4.3 Rules definition

Not only finding appropriate countermeasures, but actually implementing those that are appropriate, is essential to improve security of the service. Based on threats defined in the diagram and any other events that are monitored and one can be notified about, rules can be defined to address the scenarios we are able to foresee. Rules assume some kind of monitoring in place to provide actual value, but as long as the format can match what the SRE is able to interpret from the individual monitors, the rules are in practice agnostic to monitoring implementations.

The service tasks form the basis for defining such rules, as shown in Figure 9. The tasks are both accurate in targeting the rule's scope, yet independent of the actual service implementation we choose. Therefore, we do not need to define individual rule-sets for each potential composition plan.

In our case study service process we have already modelled a *DDoS-attack occurs on service component* threat, and this can be addressed by choosing *Threat level change* as the event type, and then selecting the service task it applies to. For values in the rule, we choose a decimal number between 0 and 1 (inclusive). This matches our threat monitoring module, which can send alerts with probability value for the escalation of threats [3]. Further, in the scope section of the rule editor, it is possible to define where in the process the rule shall apply. If a particular threat escalates for a component, we might not need to perform reactive measures unless it happens before, after or during the execution of a particular

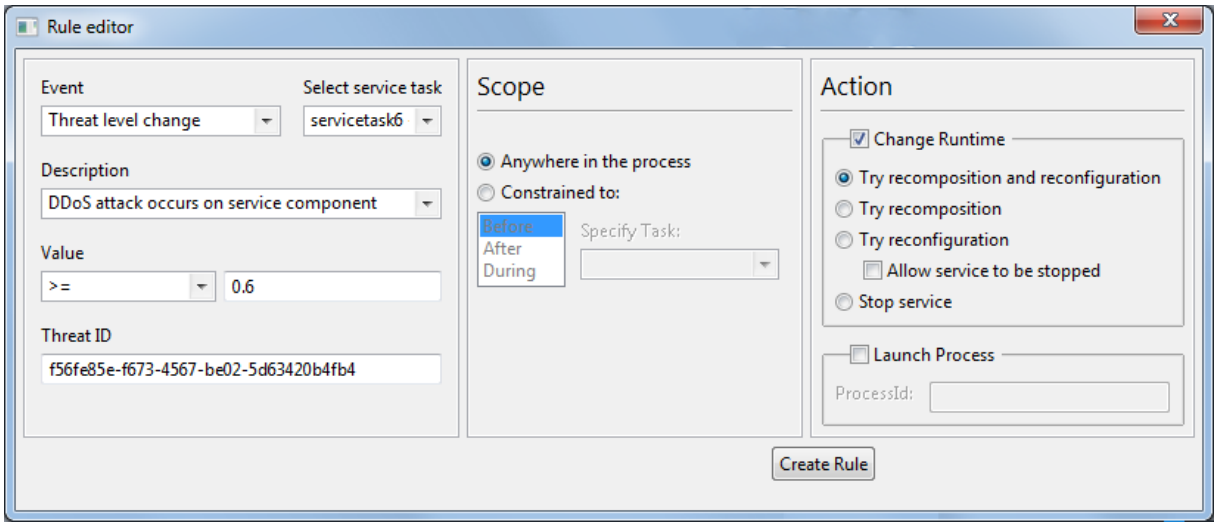

Figure 9: Threat ID appears in rule editor when threatened task is selected as basis for a new rule.

task in our process. Anyhow, we have several options for also defining the action to perform when the rule is matched, such as simply stopping the service execution (and further provision), or trying to re-compose or reconfigure the service. We also have an option to launch an additional service process, e.g. one that might initiate hardening of other parts of the system, and/or send notification messages to clients and/or service technicians. In the end, we might end up with a list of several rules for several service tasks, or simply one for the one we have defined in our case study, as shown in Figure 10.

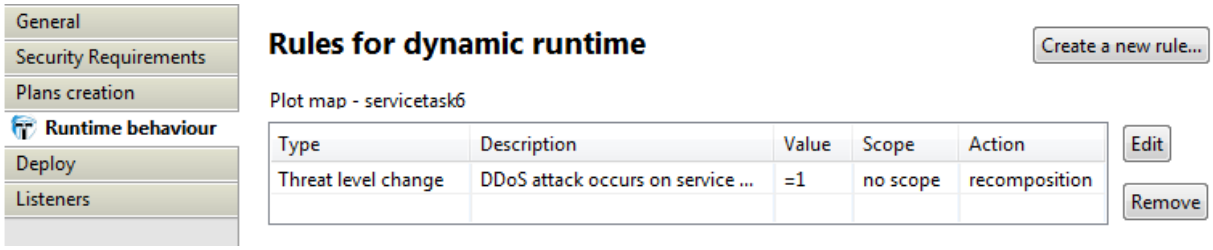

Figure 10: A rule has been defined for responding to a DDoS-attack on the map service.

2.5 Runtime management

2.5.1 Notifications

In order to operate a self-adapting service infrastructure at runtime, a commonly supported system for machine-to-machine (M2M) messaging is needed. Since there may be an endless number of services and SREs utilising this infrastructure, we cannot provide all messages to everyone. A publish-subscribe pattern is rather suitable, since we already define rules for what we need to respond to. Hence, appropriate subscriptions can be derived automatically from these rules and registered by the SRE. The SRE will then receive notifications only as specified, although the granularity of the rules will determine the relevance of e.g. slight variations in threat level probabilities (not all changes will actually trigger an action).

The SRE in our proof-of-concept creates subscriptions based on the rules which are attached to service deployment. It receives notifications according to the subscriptions, and whenever such notifications arrive, all rules are being checked for a match. The messaging system is based on Apache ActiveMQ [7], which utilises the Java Messaging Standard (JMS) to provide compatibility for many platforms and alternative protocols. ActiveMQ subscriptions are registered with a centralised broker, or network of brokers for horizontal scalability, since it is the broker(s) that deal with receiving and dispatching the notifications to all subscribers. Our broker is in addition deployed on a cloud-based infrastructure, for further increased scalability. In addition to threat level changes, we have implemented support for notifications about changes in trustworthiness, service contract violations, security properties of service components, service runtime context, and service component changes. The notifications are delivered on a best-effort basis, and may of course arrive too late in some cases (depending on how quickly a problem is discovered and duration of the attack).

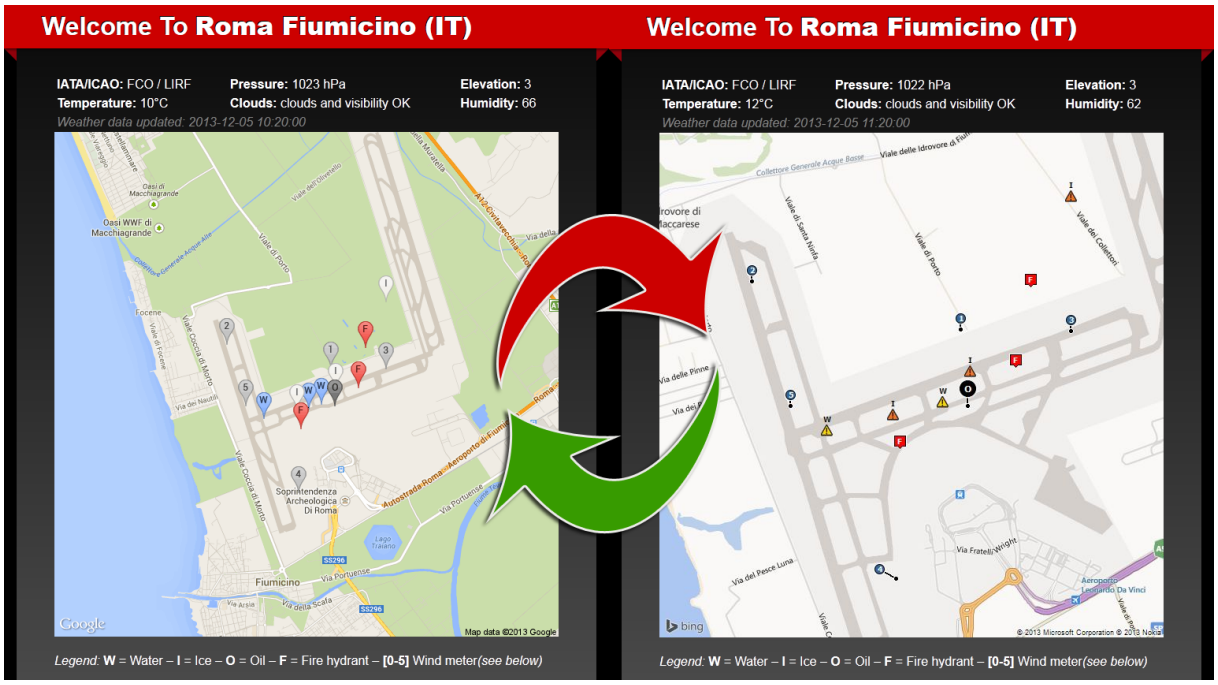

Figure 11: The SRE has replaced the Google map service with Bing Maps through a recomposition.

2.5.2 Dynamic adaptation

Since a threat monitor has now purportedly detected that the original map service used in our case study is hit by a DDoS-attack, the SRE becomes notified about this through the notification service. In order to trigger events in the self-configuring service process, the SRE must naturally be able to receive such notifications and align these with the previously defined and deployed rules. As it finds a match with the rule we defined earlier on DDoS-attacks, the SRE initiates the specified action according to that rule, which is here to try a recomposition.

Since we had support for two different map services providing the same functionality, we have prepared an additional composition plan for the airport report service. When the notification concerning a DDoS-attack on the map service is triggered and received by the SRE, the first plan no longer satisfy

our security requirements through verification [22]. Since the rule in Figure 10 provides the match, a recomposition is initiated accordingly. The original composition plan with the DDoS-ed map service will be ignored, and the second plan becomes the top-ranked one based on a chosen ranking criteria. The recomposition proceeds with deploying the second composition plan, containing the alternative map service instead of the original one. Nevertheless, the same level functionality is provided, as illustrated with Figure 11 where the airport reports, before and after recomposition, are lined up next to each other. Before the new composition is deployed, the SRE invokes a runtime verification, and the time this takes depends on the complexity of the composition. This is a well-known scalability issue associated with any kind of runtime adaptation.

3 Discussion

An obvious shortcoming to our approach is that we do not integrate a proper information security risk assessment anywhere. This is usually where threats are actually identified in a systematic manner, and where they are ranked in the order of overall risk. With this approach, we do assume that such risk assessment activities are already taking place, and that the results from these are actively used to enrich the goal and service process models. However, we also suggest that enrichment can also occur in the opposite direction, so that threat discoveries made during the requirements engineering and service design process can indeed enrich the identification phase of a risk assessment. If tool-support is to be added for this interactivity, it is indeed possible to use the threat repository service for exchanging the threat information between applications.

In terms of threats within the service process model, we have been forced to make a pragmatic compromise. Since BPMN is a process language, it is capable of defining an entire process as it is *intended* to flow, however not without language constructs for handling quite a few exceptions. Threats also provoke exceptions to normal flow, and although explicitly missing from the language, we have earlier concluded that the graphical language itself is even capable of representing what is needed here [10]. Still, we found that runtime support for handling externally triggered notifications on the process level is not handled in BPMN software tools such as the Activiti process engine. That is the reason we developed a custom plug-in for the SRE, and consequently a compatible rule editor was developed and integrated into the BPMN tool as well.

Another limitation with the tool-set we present, is that process diagrams are not supported with automatic reflection of changes in the goal diagrams. This may impact an iterative process which goes back and forth between goals and process models. Mitigating this problem through re-transformation is possible, albeit not very appealing, since any process information that is not part of the goal diagram and its security commitments will be lost. Our transformation tool is however capable of comparing any BPMN-diagram with a security requirements specification document, in order to point out nonconformity with the original security requirements. While not yet implemented, this tool could in theory be made capable of also including threats in the analysis, and point out which threats are missing in the BPMN-diagram. The problem here is however if a business-level threat from the goal model, such as “Integrity error”, has been manually sub-typed by the process designer to either one or several operational-level threats, e.g. “Checksum mismatch” or “Certificate verification failed”. Our threat repository does not express a hierarchy of the threats, and we cannot automatically trace neither the parent from a child nor the children of a parent. It is not always straightforward to create such threat hierarchies either, in particular when a child has several parents.

The main motivation for threat modelling described in this paper is related to preventive and correc-

tive measures for design-time creation and runtime execution of composite services. Another significant motivation that we have not delved into here is the need for engineers and business-people to communicate better around security when systems are designed and implemented, as pointed out by Wolter et al. [19]: *"It is evident that both security and business domain experts need to be able to define their security goals collaboratively on a common abstraction level"*. For this we believe that the use of threats in well-defined graphical modelling languages, such as STS-ml and BPMN can be a positive contribution. We currently have ongoing work on practical evaluation of the presented tools and services within the ATM domain (and others), along with other functionality on security requirements management which is out of scope for this paper. Through this evaluation we hope to gain more knowledge on what will be necessary for a broad uptake of such an approach to managing threats. Along with the already available STS-Tool [18], the described software modules will eventually be released as part of our extended BPMN tool, and/or as online services.

4 Conclusion

We have demonstrated how graphical threat modelling can benefit design-time mitigation and runtime correction of unwanted incidents. Though a service may be regarded as secure enough in the early life-cycle phases, risks are not static, and threats, vulnerabilities, probabilities and consequences can change abruptly. Tool-support is therefore essential to be able to analyse which measures to implement at design-time, and create mechanisms to handle the residual ones at runtime through e.g. automatic adaptation.

An essential part of the tool-chain we have presented is the use and re-use of threats through a shared threat repository that many tools are able to interface with. Having a common reference on the name, nature and mitigation options for the threats has been valuable to us both during the graphical modelling at design-time and for runtime alert messaging and response. By basing our work on existing languages and tools, we believe that chances of uptake is significantly larger compared to a comprehensive tool-set built from scratch. However, this also means that some transitions between the tools can be seen as a bit cumbersome. For our future work we seek to gain more knowledge on usability and integration with existing risk assessment methods that should complement the threat modelling.

5 Acknowledgement

The authors would like to thank in particular Francesco Malmignati, Balazs Kiss, Mauro Poggianella, Mattia Salnitri, Eider Iturbe and Konstantinos Giannakakis for their technical work related to enabling our proof-of-concept implementation. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no 257930 (Aniketos).

References

[1] Activiti (2013): *Activiti BPM platform website*. Available at `http://www.activiti.org/`.

[2] Aniketos project (2013): *Deliverable 5.2 Initial ANIKETOS platform integration*. Technical Report. Available at `http://www.aniketos.eu/content/deliverables`.

[3] Aniketos project (2013): *Deliverable D4.3 Algorithms for responding to changes and threats*. Technical Report. Available at `http://www.aniketos.eu/content/deliverables`.

[4] Achim Brucker & Isabelle Hang (2013): *Secure and Compliant Implementation of Business Process-Driven Systems*. In: *Business Process Management Workshops, Lecture Notes in Business Information Processing* 132, Springer Berlin Heidelberg, pp. 662–674. Available at http://dx.doi.org/10.1007/978-3-642-36285-9_66.

[5] Achim D. Brucker, Francesco Malmignati, Madjid Merabti, Qi Shi & Bo Zhou (2013): *A Framework for Secure Service Composition*. In: *ASE/IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, IEEE Computer Society, Los Alamitos, CA, USA. Available at http://www.brucker.ch/bibliography/abstract/brucker.ea-framework-2013.

[6] Eurocontrol (2014): *System Wide Information Management (SWIM)*. Available at http://www.eurocontrol.int/swim.

[7] The Apache Software Foundation (2013): *Apache ActiveMQ website*. Available at http://activemq.apache.org/.

[8] The Eclipse Foundation (2013): *Eclipse IDE downloads website*. Available at http://www.eclipse.org/downloads/.

[9] Per Håkon Meland, Shanai Ardi, Jostein Jensen, Erkuden Rios, Txus Sanchez, Nahid Shahmehri & Inger Anne Tøndel (2009): *An Architectural Foundation for Security Model Sharing and Reuse*. In: *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pp. 823–828, doi:10.1109/ARES.2009.110.

[10] Per Håkon Meland & Erlend Andreas Gjære (2012): *Representing Threats in BPMN 2.0*. In: *ARES*, IEEE Computer Society, pp. 542–550. Available at http://doi.ieeecomputersociety.org/10.1109/ARES.2012.13.

[11] Per Håkon Meland, Erlend Andreas Gjære & Stephane Paul (2013): *The Use and Usefulness of Threats in Goal-Oriented Modelling*. In: *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pp. 428–436, doi:10.1109/ARES.2013.57.

[12] E. Oladimeji, S. Supakkul & L. Chung (2006): *Security threat Modeling and Analysis: A goal-oriented approach*.

[13] OMG (2011): *Business Process Model and Notation (BPMN) Version 2.0*. Available at http://www.omg.org/spec/BPMN/2.0/.

[14] Elda Paja, Fabiano Dalpiaz, Mauro Poggianella, Pierluigi Roberti & Paolo Giorgini (2012): *STS-Tool: Using Commitments to Specify Socio-Technical Security Requirements*. In: *Advances in Conceptual Modeling, Lecture Notes in Computer Science* 7518, Springer Berlin Heidelberg, pp. 396–399. Available at http://dx.doi.org/10.1007/978-3-642-33999-8_48.

[15] Elda Paja, Fabiano Dalpiaz, Mauro Poggianella, Pierluigi Roberti & Paolo Giorgini (2013): *Specifying and Reasoning over Socio-Technical Security Requirements with STS-Tool*. In: *ER*, pp. 504–507. Available at http://dx.doi.org/10.1007/978-3-642-41924-9_45.

[16] SHIELDS project (2013): *Security Vulnerability Repository Service (SVRS)*. Available at https://svrs.shields-project.eu/SVRS/.

[17] SESAR Joint Undertaking (2014): *Single European Sky ATM Research (SESAR)*. Available at http://www.sesarju.eu.

[18] University of Trento (2013): *Website of the Socio-Technical Security modelling language and tool*. Available at http://www.sts-tool.eu/.

[19] Christian Wolter, Michael Menzel, Andreas Schaad, Philip Miseldine & Christoph Meinel (2009): *Model-driven business process security requirement specification*. *Journal of Systems Architecture* 55(4), pp. 211–223.

[20] World Wide Web Consortium (W3C) (2013): *Web Services Description Language (WSDL) 1.1*. Available at http://www.w3.org/TR/wsdl.

[21] Bo Zhou, David Llewellyn-Jones, Qi Shi, Muhammad Asim & Madjid Merabti (2013): *Prototype for design-time secure and trustworthy service composition*. In: *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pp. 847–848, doi:10.1109/CCNC.2013.6488561.

[22] Bo Zhou, David Llewellyn-Jones, Qi Shi, Muhammad Asim, Madjid Merabti & David Lamb (2012): *Secure Service Composition Adaptation Based on Simulated Annealing*. In: *6 th Layered Assurance Workshop*, p. 49.

Towards the Model-Driven Engineering of Secure yet Safe Embedded Systems

Ludovic Apvrille
Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI
Sophia Antipolis, France
ludovic.apvrille@telecom-paristech.fr

Yves Roudier
EURECOM
Sophia Antipolis, France
yves.roudier@eurecom.fr

We introduce SysML-Sec, a SysML-based Model-Driven Engineering environment aimed at fostering the collaboration between system designers and security experts at all methodological stages of the development of an embedded system. A central issue in the design of an embedded system is the definition of the hardware/software partitioning of the architecture of the system, which should take place as early as possible. SysML-Sec aims to extend the relevance of this analysis through the integration of security requirements and threats. In particular, we propose an agile methodology whose aim is to assess early on the impact of the security requirements and of the security mechanisms designed to satisfy them over the safety of the system. Security concerns are captured in a component-centric manner through existing SysML diagrams with only minimal extensions. After the requirements captured are derived into security and cryptographic mechanisms, security properties can be formally verified over this design. To perform the latter, model transformation techniques are implemented in the SysML-Sec toolchain TTool in order to derive a ProVerif specification from the SysML models. An automotive firmware flashing procedure serves as a guiding example throughout our presentation.

1 Introduction

Most contributions around Model Driven Engineering (MDE) now offer appropriate methodologies and modeling environments for designing safe, complex, distributed, and real-time embedded systems. The analysis of timing constraints, scheduling, resource allocation, and concurrency are commonly handled by these environments. In contrast, security has long been considered in retrospect, especially after serious flaws were discovered in computerized systems. Security as well as privacy issues have in particular only recently become a major concern in embedded systems. However, the size, heterogeneity, and communication features of modern embedded systems make it compelling to develop a suitable engineering environment to more explicitly define security objectives and threats, to implement countermeasures with security mechanisms, and to assess or even formally prove the effectiveness of security countermeasures.

We introduce SysML-Sec, a SysML-based Model-Driven Engineering environment aimed at fostering the collaboration between system designers and security experts at all methodological stages of the development of an embedded system. SysML-Sec introduces both customized SysML diagrams for security matters and an associated methodology. The SysML-Sec methodology includes three SysML-based stages. (i) System analysis starts with a partitioning-based process in which security requirements and threats can be identified together with functional features of the system. (ii) System design focuses on software-implemented security mechanisms. Finally, (iii) System validation intends to formally verify, simulate, and test the models built at previous stages by relying on model transformation techniques. This paper presents the overall methodology, with a particular focus on the design and proof of security mechanisms.

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

The SysML-Sec methodology and diagrams have been developed and experimented in the scope of the FP7 European project EVITA, which resulted in the design and implementation of a secure architecture for automotive embedded systems. The definition, design, and validation of this architecture was performed with the methodology that is presented in this paper. Thus, more than 20 use cases (notably an emergency braking use case) were taken into account for that purpose. The diagrams in this paper are directly excerpted from the EVITA "firmware flashing" case study.

2 Context: embedded systems

2.1 Designing embedded systems

IT systems are commonly designed following a V-cycle, with building stages (requirements, analysis, design, deployment) followed with verification stages (e.g., tests, formal proofs). For embedded systems, the V-cycle can obviously start only once functions have been partitioned into software and hardware. System partitioning usually relies on the Y-chart approach [7]. This is the very first step to co-design software and hardware functions on the one hand, and the hardware architecture (defined in terms of execution, communications, and storage) on the other hand. The result of this process is an optimal hardware / software architecture with regards to criteria at stake for that particular system (e.g., cost, performance, etc.). In the scope of the DIPLODOCUS environment [4], on top of which SysML-Sec is built, the Y-Chart is implemented as follows:

1. *Applications* are first described as abstract communicating tasks: tasks represent functions independently from their implementation form.
2. Hardware *architectures* are described as a set of abstract execution nodes (e.g., CPU with operating systems and middleware, hardware accelerators), communication nodes (e.g., buses), and storage nodes (e.g., memories).
3. A *mapping* model [7] defines how tasks and communications between tasks are assigned to computation and communication / storage elements, respectively. Tasks are also partitioned between hardware and software. For example, a task mapped on a hardware accelerator is a hardware-implemented function whereas a task mapped over a CPU is a software implemented function.

This partitioning process is of utmost importance. Indeed, if critical high-level design choices are invalidated afterwards because of late discovery of issues (performance, power, etc.), then it may induce prohibitive re-engineering costs and late market availability.
DIPLODOCUS relies on the SysML allocation mechanisms for the mapping stage. The UML deployment diagram might be a good candidate for this, but it is rather used for the deployment of already-designed software functions: "the assignment of software artifacts to nodes", as stated in the UML standard. In DIPLODOCUS, functions may be fully hardware implemented. Moreover, they are highly abstracted, that is, we do not map any concrete artifacts (e.g., a source file), but only high-level functional elements.

2.2 Security issues in embedded systems

An increasing number of embedded systems have become communicating artifacts, feature new interactions with their immediate environment or with backend systems, and are thus exposed to criminals. For example, attacks have been shown to be possible on set-top boxes like Microsoft's XBox [16] or ADSL routers [6], mobile appliances [13], avionics [34], or automotive systems [15] to cite but a few. Many

of these security issues reflect either the exploitation of low-level vulnerabilities, which might often be addressed with appropriate programming practices and specific component tests, or design flaws due to an insufficient understanding of the mapping of functional or security logical components to the hardware architecture. We claim that the SysML-Sec Model-Driven Engineering approach makes it possible to perform an appropriate system analysis, design and proof in both directions, and to describe both security threats and security objectives and to further prove whether their are well handled at system design.

3 SysML-Sec: an Overview

3.1 Rationale

We designed the SysML-Sec environment in order to make it possible to describe security issues together with partitioning requirements, as further discussed in [30]. In particular, our extensions bridge the gap between goal-oriented descriptions of security requirements and attacks, and the fine-grained representation of assets based on the software / hardware architecture (and their model-driven analysis). SysML-Sec also supports phases of the V-cycle after the partitioning stage, and notably the design of the software-partitioned functions[1]. The main objectives of SysML-Sec are:

- Guiding and increasing the collaboration between system engineers and security experts throughout the entire embedded system lifecycle. This has been the reason for our adoption of the OMG standards, and more specifically SysML, which are quite widespread in the embedded system world today.
- Providing detailed representations of the security threats and security requirements compatible with the MDE methodology used and making it possible to adopt a stepwise refinement approach to the definition of both the functional and the security architecture. This refinement should also make it possible to bridge the gap between initial high-level requirements and the definition of precise and detailed security mechanisms later on.
- Combining software/hardware codesign together with the handling of security concerns. We contend that this particular design objective is a key in the embedded system domain.
- Offering simulation and formal verification capabilities at system partitioning, system design, which constitute two critical phases of embedded system engineering. At the system partitioning level, simulations help assess the impact of security features on the system performance, e.g., the impact of a security protocol on a bus load. At the system design level, formal verification intends to prove whether threats are correctly handled.

3.2 Methodology

The SysML-Sec methodology adopts a three-phase approach that first deals with the system analysis, then with software design, and finally with system validation, as depicted in Figure 1. The analysis includes the elicitation of requirements and attacks, and the partitioning of the system. The design stage includes the definition of security mechanisms, and the refinement of security requirements in security properties to be proved in the design. Lastly, the verification takes place at different engineering phases, as follows. Simulation is mostly used at the partitioning stage in order to evaluate the impact of security

[1] SysML-Sec does not address hardware design beyond the partitioning stage

mechanisms in terms or performance. Formal verification intends to prove the resilience of the system under design to threats. Testing is meant to do the same, but on the deployed implementation resulting from the design models.

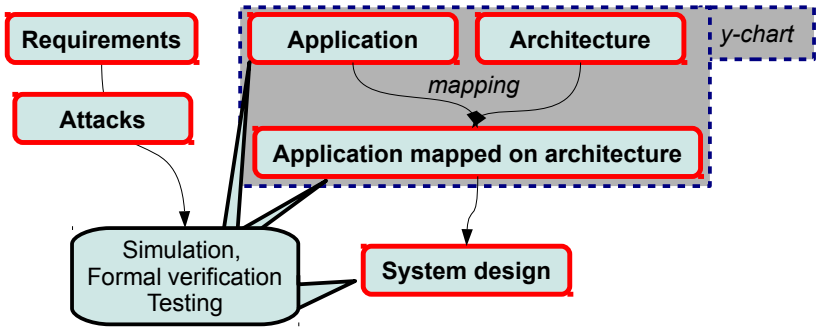

Figure 1: SysML-Sec methodology

3.3 Tooling

TTool [1] is a free software supporting several UML profiles, e.g., DIPLODOCUS and AVATAR. For the partitioning stage, SysML-Sec reuses DIPLODOCUS, as explained in [27]. Requirements, attacks, and design are analyzed with AVATAR, a profile dedicated to the analysis and modeling of embedded systems [3].

4 Case study: Firmware update

We illustrate SysML-Sec methodological stages with a ”firmware update” case study. The latter is taken from a public deliverable of the European project EVITA [20]. The main purpose of EVITA is to define a secure architecture for automotive embedded systems. Such systems are in charge of critical functions and are quite complex: they contain around 100 Electronic Controls Units (ECUs) all interconnected to the main system bus (CAN, FlexRay). Attacks are motivated by safety and economical reasons (notably theft). Security concerns, which were mostly related with physical access to the car are also evolving today with the advent of the connected car and of Car2X communications, which further exposes these systems.

The considered case study aims at the update of an ECU software with a newer version of the firmware, as performed at a service station through diagnosis tools. Car diagnosis is hardwired according to the Standard Unified Diagnosis Services UDS, which is specified in the ISO 14229-1. At first, the ECU to be flashed initializes its software and starts the diagnosis function. The service station employee connects his diagnosis tool to the on-board diagnosis interface in the vehicle. Once the diagnosis has been performed and authentication is performed, a programming session is settled. Finally, the flashing can be performed in ROM.

The overall process is expected to be secure, in particular with respect to integrity, confidentiality, and authenticity of the firmware and/or of the flashing process. More specifically, the following security requirements can be defined (see [20] for further details):

- **Authenticity**: is the vehicle sure to communicate with a valid diagnosis Tool?

- **Confidentiality**: the firmware constitutes intellectual property to protect.
- **Data integrity**: the flashed code must not have been modified.
- **Anonymity**: Private information about the driver should not be disclosed to the service station during the flashing process.

5 System requirement engineering and analysis

The security requirement and threat analysis is mostly regarded as a preamble to risk analysis in IT systems. This process is generally meant to decide whether to introduce security countermeasures into the system, which means additional costs. In the case of embedded systems, we contend that the security analysis also has a strong impact on the system architecture and its realtime performance: the security requirements and threat analysis should thus be performed along an iterative, and therefore more agile partitioning process.

5.1 Iterative security/system codesign process

System partitioning, security requirements, and threats are progressively refined based on one or several typical use cases. The following phases, which thus start with an initial architecture, are iterated in order to reach a satisfactory level of refinement:

Initial architecture mapping. The functionalities of the system highlighted in these use cases are first modeled as tasks. Exchanges between functions are modeled with information and event flows between tasks. Tasks and communications can then be mapped to a draft architecture of the system. The designer's experience plays a key role in determining the first draft of the architecture.

Architecture analysis. *System assets* are identified among architectural elements (processors, pieces of software, sensors, hardware accelerators, communication channels) and will first refer to generic components, like for example: "all system buses". When the architecture gets more detailed, assets are more likely to be refined into specific elements. The hardware/software partitioning and the function mapping adopted play a key role here in defining the type of asset at hand (and later on its vulnerabilities).

Security concern identification. *Threats and security vulnerabilities of the selected assets* should as much as possible describe the capabilities that an attacker should meet or exceed and the origin of attacks (local, remote, through a specific interface). The SysML-Sec environment supports the assessment of risks following the approach described in more detail in the EVITA case study [31, 14]. We also implemented automated checks of the threat coverage by security objectives. Based on the risk analysis, one should also identify and prioritize security objectives that are mapped to a threat.

Security objectives might originate (1) from security standards or properties expected from the system, or (2) from unaddressed threats or attacks on assets, or (3) from the refinement of another security objective when the process is iterated and the level of detail of the architecture has changed. In further iterations, one may need to update security objectives deprecated by changes in the architecture.

Architecture refinement. The architecture refinement originates from a more detailed description of the architecture components as the system and its usage become more precisely known (e.g., new communication channels, refinement of an execution environment into OS/middleware/application layers, etc.). It may also result from transitively mapping requirements to system information flows, which are often distributed among multiple hardware elements. The refinement phase may fail if the architecture and security requirements are incompatible, for instance, if the performance overhead of security mechanisms is too high. Consistency checks should also be performed to ensure that a security objective

does not conflict with another requirement expressed over the same asset. A failure is the sign that the analysis should be backtracked to the previous stage of refinement.

5.2 Diagrams

5.2.1 Requirements

Security requirements are modeled in SysML Requirement Diagrams (RD). The main operators of RDs are *Requirement Containment* and *Derive Dependency* formalisms used to define relationships between requirements. The *containment* relationship depicts sub-requirements in terms of hierarchy and enables a complex requirement to be decomposed into its containing child requirements whereas *deriveReqt* determines the multiple derived requirements that support a source requirement. A *Security Requirement* stereotype is introduced to make a clear distinction between functional requirements and security requirements of the system, yet modeling both functional and non-functional requirements in a single environment. Furthermore, a *Kind* parameter is defined to specify the category of the security requirement (*confidentiality*, *access control*, *integrity*, *freshness*, etc.).

Figure 2 represents 5 security requirements for a "Firmware update": the authenticity of the firmware, controlled access to the flash memory, itself derived into controlled access to both the flashing function and to reading the flash, and the confidentiality of firmware data.

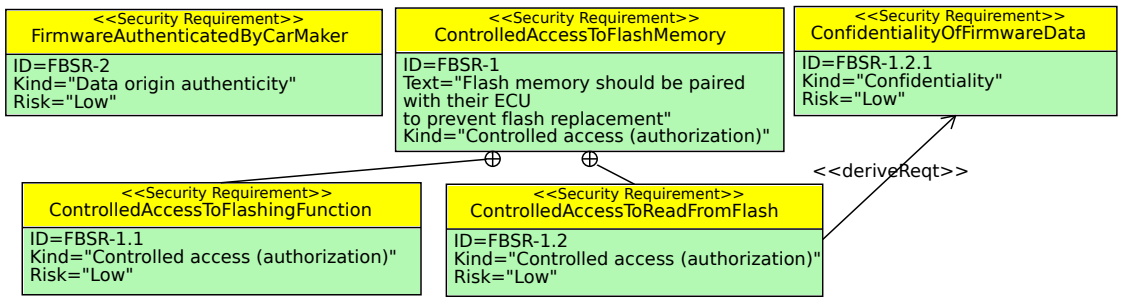

Figure 2: Excerpt from the SysML-Sec Requirement Diagram of "Firmware Update"

5.2.2 Threats and Attacks

Instead of using the traditional attack tree approach [32], we suggest that threats can be better modeled with a more relational approach, using slightly customized SysML Parametric Diagrams. Threats are modeled as values embedded into blocks representing the target of the attacks, thus achieving a representation that is more compact and better mapped to the system architecture. Attacks ($<< attack >>$ stereotype) can be linked together with logical operators like *OR*, *AND*, as well as temporal causality operators like $SEQUENCE$, $BEFORE$, or $AFTER$. We consider the latter constructs as especially helpful to describe the attacker's operational point of view in embedded systems, like for instance situations in which there is a maximum duration between two causally related attacks. For example, when attacking a system with time-limited authentication tokens, the token must be first retrieved, and then the use of this token must occur before its expiration.

Attack instances in different parametric diagrams can be linked together in order to assess the impact of a specific vulnerability and the need to address it at the risk assessment phase. An attack can also be

tagged as a *root* attack, meaning that this attack is at the top of a tree of attacks. Last but not least, attacks can be linked to requirements, thus allowing an automated check of the coverage of attacks.

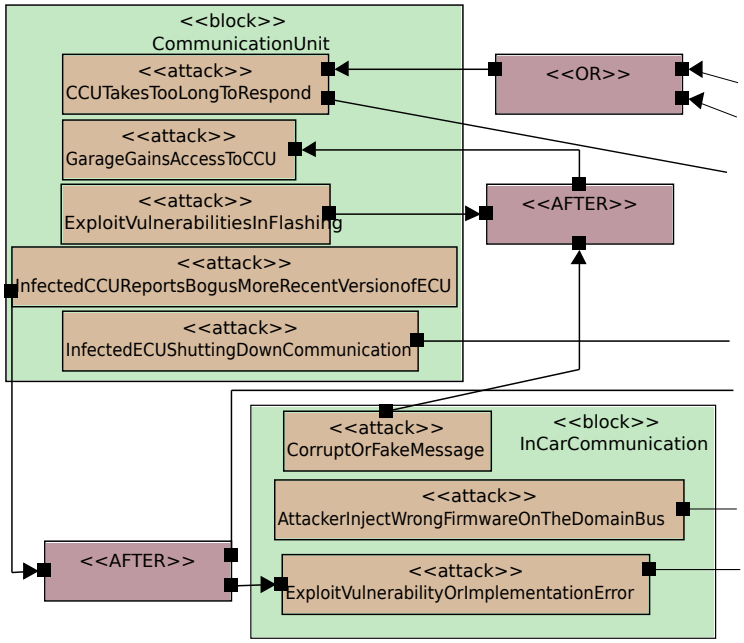

Figure 3: Excerpt from the Attack Diagram of ”Firmware Update”

Figure 3 depicts a few attacks identified in the scope of the EVITA ”Firmware update” case study. Two assets are represented (*CommunicationUnit*, *InCarCommunication*). For example, performing the attack ”ExploitVulnerabilityinFlashing” in *CommunicationUnit* and then forging a ”CorruptOrFakeMessage” in the *InCarCommunication* makes it possible to perform ”GarageGainsAccessToCCU”.

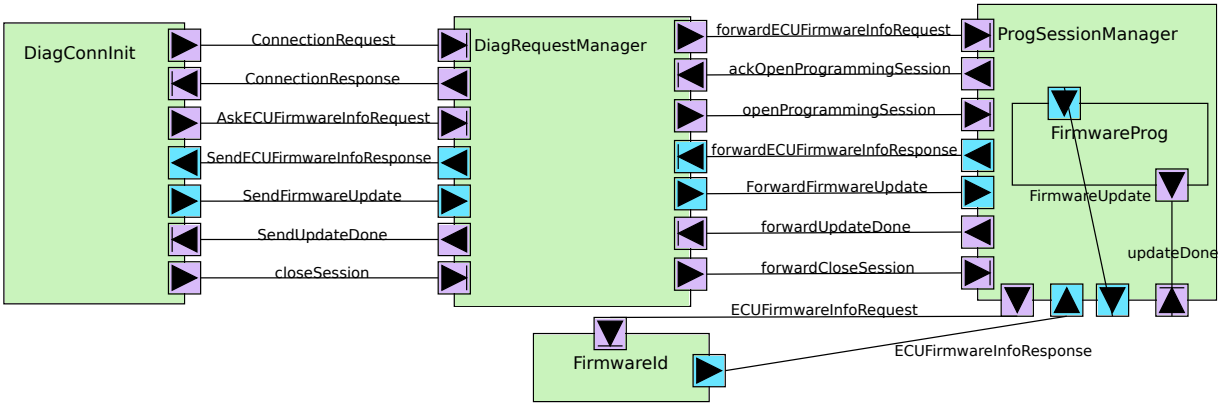

Figure 4: Application model of firmware update

5.2.3 Partitioning

The application model is a graph of communicating functions. For example, in Figure 4, event flows correspond to purple ports, and information-flows to blue ports. Five main functions are modeled: the initialization of the diagnosis connection (*DiagConnInit*), diagnosis request management (*DiagRequestManager*), programming session management (*ProgSessionManager*), firmware identification (*FirmwareId*) and lastly, the firmware programming (*FirmwareProg*).

The architecture model is partially depicted in Figure 5. It represents the assets to be protected. Two subdomains are connected to the main CAN bus. Each subdomain has its own CPU, RAM, and bus. The first domain also has a hardware accelerator, and the second one has a flash memory that the procedure intends to update.

The mapping consists in assigning functional elements to assets, i.e., assigning tasks to either a CPU or hardware accelerators, and communications to buses and memories. Figure 5 displays the mapping of three tasks in the two subdomains: *ProgSessionManager* is mapped to "CPU_PTC" and *FirmwareProg* and *FirmwareId* are mapped on "CPU_ECU".

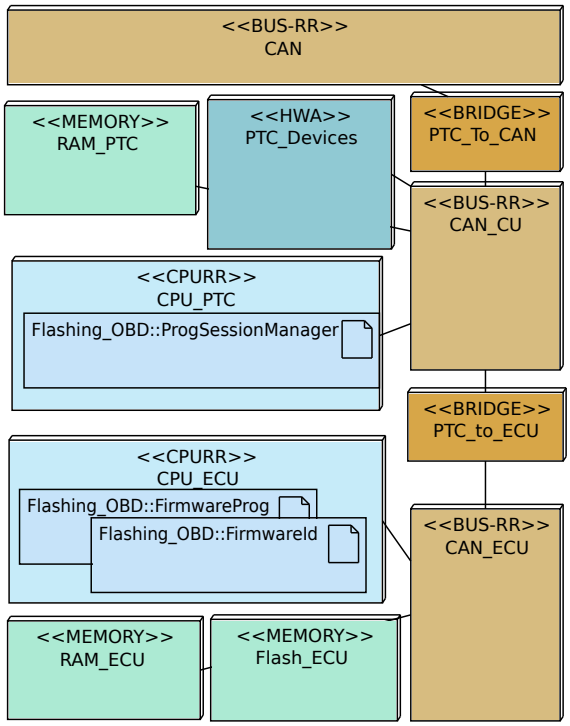

Figure 5: Excerpt from the architecture and mapping model of firmware update

6 System design

6.1 Methodological aspects

System design intends to refine the architecture and behavior of all functions mapped over processor nodes during the partitioning. From a security point of view, the aim is to describe in more detail how

security requirements can be fulfilled with security hardware mechanisms of software mechanisms executed on top of the hardware architecture defined in the partitioning stage, and to verify that requirements identified during the partitioning phase are really satisfied by this design. Requirements expressed at partitioning are informal and refer to assets: they therefore need to be refined until their expression directly relates to design elements (e.g., attributes, methods, exchanged messages, states, etc.). Once refined, they constitute the security properties that are to be verified, notably through formal methods.

Diagrams of SysML generally used for system design, like block diagrams and state machine diagram, lack explicit ways to model security mechanisms. For example, security mechanisms commonly need the pre-sharing of cryptographic material (e.g., secret keys), but block diagrams have no way to model this. Thus, SysML-Sec extends SysML in order to explicitly model security mechanisms and properties.

6.2 Security design extensions

A SysML-Sec design is made upon SysML block and state machine diagrams, extended with several features, and formally defined in pi-calculus (a process algebra). We assume a Dolev-Yao attacker model, that is, only messages exchanged between blocks can be eavesdropped, contrary to attributes of blocks that are considered as private from the point of view of the attacker. That attacker model is enough to describe attacks on the protocols deployed between the components of the embedded system, from outside (i.e., using communication networks) or from within the system (i.e., using internal buses or any other accessible component interface within the system). It however does not aim at capturing physical attacks on the hardware, nor a sequence of exploitation of vulnerabilities of several components. The main extensions are:

- **Public and private channels**: Since communication channels may have been mapped over secure or non secure buses at the partitioning stage, we give the possibility to tag links between blocks with a *public* label if an attacker can eavesdrop, or with a *private* label otherwise.
- **Cryptographic algorithms**. SysML-Sec blocks can define a set of methods corresponding to cryptographic algorithms (e.g., *encrypt*), *verifyMAC*, etc. so as to be able to describe security mechanisms built upon these algorithms, e.g., cryptographic protocols.
- **Cryptographic material**. Blocks can also pre-share values, a feature commonly needed to setup cryptographic protocols. SysML-Sec introduces specific *pragmas* for that purpose: (*InitialSystemKnowledge* and *InitialSessionKnowledge*). The first one is used to describe data that are shared before the system execution. The second one defines data that are shared within each session of the same system. Typically, when considering a cryptographic protocol, the first pragma means that the data are pre-shared and common to all protocol sessions, and the second one states that the data are shared but have different values in each protocol session.

  ♯ `InitialSystemKnowledge BlockID.attribute` [`BlockID.attribute`]*

For example, Figure 6 displays the block diagram of a key distribution protocol used as a security protocol so as to allow ECUs to communicate in a secure way. That protocol is built upon three different entities: the initiator of the key distribution (*ECU1*), a key master (*KM*) and the ECUs with which ECU1 expects to share keys (*ECUN*). All blocks define cryptographic functions, two of them being visible in ECUN (*encrypt()*, *decrypt()*). The pragma "InitialSystemKnowledge" states that *PSK1* is an attribute which is shared between ECU1 and KM before system startup.

Figure 6 presents the subset of the KM state machine diagram. State machines are used to describe the behavior of each block. They can manipulate cryptographic functions and data. For example in the state

machine of the KM block, the action $msg8 = MAC(msg1, PSK1)$ assigns the result of $MAC(msg1, PSK1)$ to $msg8$.

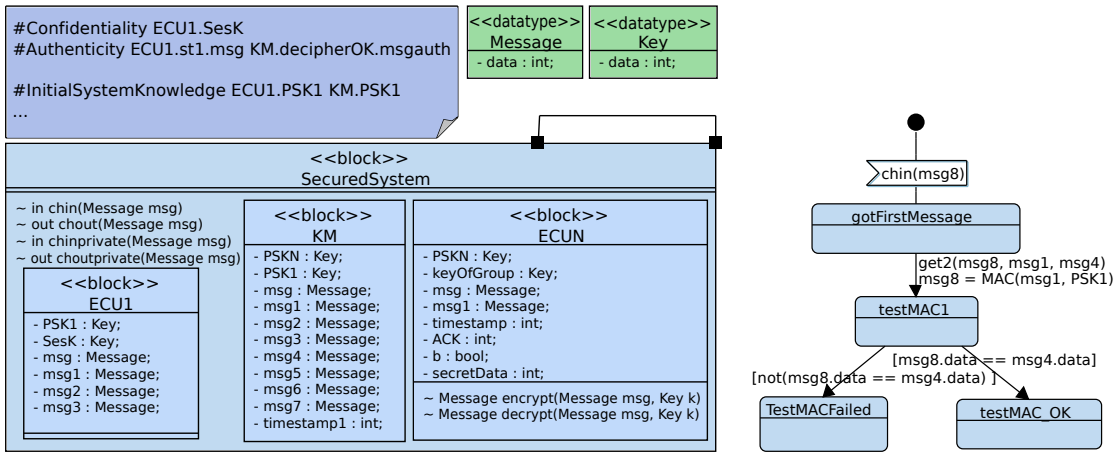

Figure 6: SysML-Sec block diagram and a state machine diagram of Key Distribution Protocol

6.3 Security properties

Security properties linked to the design are obtained from a refinement of security requirements elicited at analysis stage. A dedicated language has been defined for describing the commonly complex safety properties, which is based on SysML Parametric diagrams [22]. On the contrary, security properties can usually be defined with a type (e.g., *confidentiality*), and with design elements related to that kind (e.g., the confidentiality of the attribute of a block). This simplicity pleads for a basic modeling solution, that is not based on complex diagrams or operators. Our solution again relies on *pragmas*, provided in notes of Block Diagrams: *confidentiality* and *authenticity* can be directly expressed at this level.

A confidentiality pragma states that the content of an attribute of a block shall never be disclosed to an attacker:

```
♯ Confidentiality block.attribute
```

An authenticity pragma states that a message $m2$ received by a block $block2$ was necessarily sent before by block $block1$ in a message $m1$. The following examples describes such a situation:

```
♯ Authenticity block1.s1.m1 block2.s2.m2
```

This authenticity pragma specifies two states: one associated with the sender block, and one associated with the receiving block, i.e. one state $s1$ in $block1$, and one state $s2$ in $block2$. Also, in the state machine diagram of $block1$, $s1$ corresponds to the state right before the sending of $m1$. Analogously, $s2$ corresponds to the state right after message $m2$ has been received and accepted as authentic.

The confidentiality pragma of Figure 6 states that the PSK1 (pre-shared key 1) of ECU1 shall never be disclosed (secret key). This property is derived from the requirement on confidentiality: secret keys must remain secret so as to ensure the confidentiality of the firmware that is to be sent to the flash memory. Similarly, the authenticity pragma states that *msgauth* received by KM right after state *decipherOK* has necessarily been sent by ECU in variable *msg* right before state *st1*.

7 System validation

A system validation can be performed from partitioning models (e.g., performance evaluation of the selected hardware architecture: load of CPUs and buses), from design models (e.g., proof of safety and security properties), or from executable code automatically generated from deployment models (e.g., safety and security tests) [2]. Model transformations have been defined in order to transform SysML-Sec models into simulation, formal, or executable specifications. Mapping model transformations are given in [21], and design model transformations are described in [26] and [17].

From partitioning models, it is possible to evaluate the impact of security mechanisms onto safety constraints, e.g., real-time constraints such as latencies. An example of such a study is given in [33] where the impact of adding security-oriented MAC information in the messages of CAN bus is checked against safety-oriented properties (latencies) at partitioning stage. The impact of security mechanisms onto the load of buses and CPUs might be evaluated as well, as they might impact the behaviour of safety-critical processes and communications. For instance, the decryption of packets sent from an ECU to another one may prevent the scheduling of the process controlling the brake actuation.

From SysML-Sec designs, the formal proof relies respectively on UPPAAL [8] and on *ProVerif* [10] for the proof of safety and security properties (see Figure 7). Our safety proofs take into account all design elements, including security mechanisms like the liveness and reachability of all states of cryptographic protocols, and the impact of security mechanisms onto safety-critical processes. Our security proofs may relate to any behavior described with state machines. The notion of protocol is central here: it captures both communications over buses or networks and communications between components in separate blocks like separate trust zones within a processing unit, for instance when virtualization mechanisms are used.

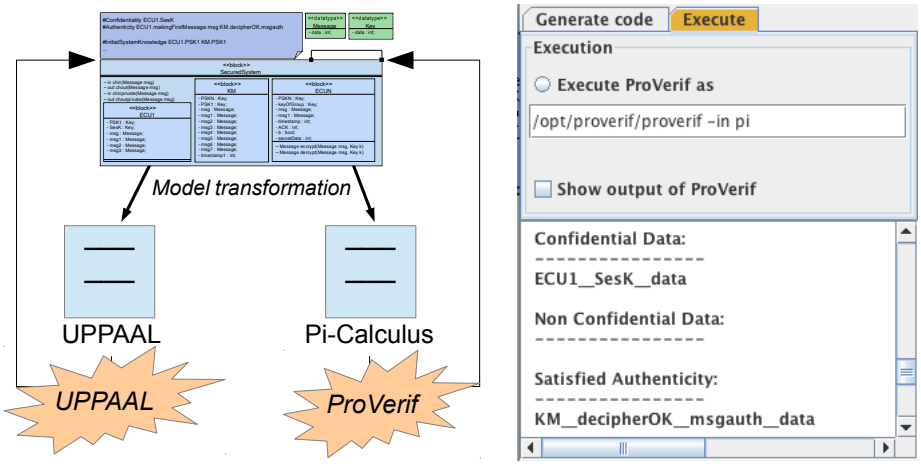

Figure 7: (a) Model transformations of TTool for proving safety and security properties. (b) TTool assistant for the formal verification of confidentiality and authenticity properties defined at Figure 6

ProVerif [10] is a toolkit that relies on Horn clauses resolution for the automated analysis of security properties over cryptographic protocols, under the Dolev-Yao model. ProVerif takes in input a set of Horn Clauses, or a specification in pi-calculus together with a set of queries. ProVerif outputs whether each query is satisfied or not. In the latter case, ProVerif tries to identify a trace explaining how it came to the conclusion that a query is not satisfied. Figure 7 depicts the successful verification in TTool of the confidentiality and authenticity properties modeled in Figure 6. While we can specify any security

requirement, we currently only support the formal validation of those that can be expressed with these two security properties.

Safety proofs take into account all design elements apart from the ones specifically defined for security purposes, that is the security-oriented pragmas and cryptographic methods that have no impact on safety properties (liveness, reachability). Similarly, the proof of security properties abstracts away irrelevant system details. Such a proof does not require values on variables nor temporal information. Several state machine elements, like temporal operators (*after* clause) and computations with variable values, are not taken into account for the security proof. For instance, $a = b + c$ is executed in a symbolic way, but without concrete values: the data flow captured in that manner (we know that a contains information from b and c) is the only useful information for security proofs.

8 Related work and perspectives

Many proposals already exist that address security requirements engineering or threat modeling in IT systems. In [24], Nhlabatsi et al. classify security requirements engineering work according to four dimensions, namely: *goal-based approaches*, *Model-based approaches*, *problem-oriented approaches*, and *Process-oriented approaches*. Our approach combines a goal-based description of security requirements with a model-driven engineering of the system architecture and threat analysis, and emphasizes the importance of agile interactions throughout the engineering V-cycle. In that respect, its philosophy follows that of the TwinPeaks approach [25], even though the latter does not address hardware systems. Instead of a simple spiral alternating between the requirements and the architecture as TwinPeaks suggests, we also alternate between the Y-Chart modelling of software and its mapping to hardware components, the identification of assets and threats to them, and the identification of security requirements.

Assessing security properties at design level mostly relies on formal approaches. For example, [35] proposes to verify cryptographic protocols with a probabilistic analysis approach. In more recent efforts, [11] introduces a first order Linear Temporal Logic (LTL) into the Isabelle/HOL theorem prover, thus making it possible to model both a system and its security properties, but unfortunately leading to non-easily reusable specific models. [23] mixes formal and informal security properties, but the overall verification process is not completely automated, again requiring specific skills. Our approach focuses on refining semi-formal specifications into formal models using a graphical language, with the aim of formal validation of the satisfaction of security properties. In this respect, it may seem very similar to the UMLsec approach [19], a modeling framework aimed at defining security properties of software. UMLsec also primarily features a model-based approach to security requirements engineering according to the abovementioned classification. It makes it possible to define security properties of software components and of their composition within a UML framework. It also features a rather complete framework addressing various stage of model-driven secure software engineering from the specification of security requirements to tests, including logic-based formal verifications regarding the composition of software components. With respect to the embedded system field which the current paper focuses on, UMLsec allows for the description of the mapping of already-designed software onto hardware nodes [19] through the use of UML deployment diagrams. According to OMG's own definition, such diagrams describe "a set of constructs that can be used to define the execution architecture of systems that represent the assignment of software artifacts to nodes.". This means that they unfortunately cannot handle software-hardware partitioning in the sense of [7]. In contrast, SysML-Sec adopts the more holistic point of view of SysML proponents and focuses on the very evaluation of various mappings for security and functional features before and during detailed system design. Additionally, SysML-Sec provides hardware

nodes with an explicit autonomous simulation and proof semantics. Lastly, SysML-Sec includes low-level hardware nodes whose use may impact communications and executions of functions, e.,g. a DMA engine may favorize, or not, a urgent communication with regards to security-oriented communications. Similarly to UMLsec[18], our proposal additionally features a goal-based approach to describe security objectives. Still, our approach fits entirely within SysML with minor extensions (essentially new stereotypes and a few operators) and does not introduce any new diagram in order to make it easy to adopt for system engineers.

From our experience, partitioning is in our opinion a central issue in embedded systems. Achieving a correct partitioning that adheres to safety requirements necessitates that the impact of security mechanisms be understood and quantified as early as possible. We note that only a few authors, notably Eames and Moffet [12], and more recently Pièvre-Cambacédès and Bouissou [28] and Raspotnig and Opdahl [29] have dealt with the relationships between security and safety requirements. For instance, the last two authors discuss quite thoroughly the relationships that can be established between security and safety requirements. In particular, these studies can describe conflicts, like we considered in the current paper, but also reinforcement relationships (when safety and security concur towards the same design), or conditional dependence. We think that obtaining similar descriptions within SysML-Sec would require the engineering methodology to be extended with an additional feedback interaction from all engineering phases to the specification phase: for instance, the satisfaction of safety requirements should be checked based on the security mechanisms introduced before any further safety mechanism would be introduced. We did not evaluate any such methodological step in our case study, which did not feature safety requirements. However, we plan to investigate such issues in the future.

The work we presented in this paper aims more specifically at validating the innocuity of security mechanisms with respect to safety requirements whose specification is outside the scope of this paper. This is based on quantitative validations during the specification and design phases, together with the refinement of the system architecture. In particular, approximations can first be made based on the nature of security mechanisms, and most notably, the cryptographic algorithms used and their costs (both computationally and in terms of bandwidth usage). As the design of the security mechanisms to be deployed is increasingly refined, together with a more detailed hardware/software partitioning, simulations by the SysML-Sec framework can be used to achieve a more precise evaluation of those elements. Although not yet supported by our toolchain, tests might finally be used on the developed components to validate the adherence to safety requirements. Threat analysis itself pleads for such a move, to incorporate the description of security tests performed over implementations into the system model. In this respect, our proposal includes the use of the SysML parametric diagram for threat modeling. SysML-Sec can thus encode an attack tree, yet it adopts a block-centric perspective with reuse in mind. We especially think that this will allow for the composition of the threat modeling performed by security analysts about commercial off-the-shelf (COTS) components with system specific analyses. We plan to further extend SysML-Sec expressivity here: our declarative approach should be especially useful in order to incorporate knowledge from other threat modeling approaches.

Tool support is in our opinion critical to the successful and systematic engineering of systems, and especially so in what regards non-functional requirements like security and the validation of their satisfaction in a given design. This need has been recognized by many other authors at various levels of the development lifecycle of IT systems. Furthermore, the availability of free software is another important factor of success, would it only be due to their extensibility for specific needs. The KAOS framework [36], which pioneered goal-oriented security requirements engineering, features an entire dedicated environment. CARiSMA (a recent extension of UMLsec) integrates into the Eclipse development environment, which ensures a better acceptance factor. Threat modeling frameworks have also been developed

like Isograph Software's AttackTree+ for designing attack trees or the free software ADTool [9]. There also exists a number of formal verification tools with nice interfaces like AVISPA [5], which provides good support for protocol verification. It can be observed that many of these tools address only a part of the development lifecycle of a system, which is detrimental to their adoption. Our framework aims at a more comprehensive environment adhering to SysML, a formalism supported by the OMG and INCOSE, and widely known to system engineers.

9 Conclusion and future work

Embedded systems are becoming ever more complex and intricate combinations of software and hardware, exposed to a plethora of attacks as those systems open up to attackers through the introduction of communication functionalities. The economic incentives to reduce the time-to-market of such systems while ensuring equally good or better engineering qualities require the introduction of systematic security engineering methodologies with an appropriate support. The adoption of user-friendly tools featuring automated and simplified verification will likely become mandatory in this respect. Our proposal, SysML-Sec, specifically addresses that need at the diverse phases of system design and development. It is based on a popular language (OMG's SysML) and supported by a free software toolkit (TTool). It features formal verification capabilities that rely on recognized toolkits for security (ProVerif) and safety (UPPAAL). The methodology we developed has been experimented to define a secure automotive embedded system. We presented the results of a case study on that system, which highlights several advantages of our approach, especially regarding formal verification. Our future work will be dedicated to further evaluating our approach on other embedded systems as well as to extending the tool support, especially for enhancing threat coverage and requirements traceability analyses.

References

[1] L. Apvrille (2013): *TTool website*. In: *http://ttool.telecom-paristech.fr/*.

[2] L. Apvrille & A. Becoulet (2012): *Prototyping an Embedded Automotive System from its UML/SysML Models*. In: *ERTSS'2012*, Toulouse, France.

[3] L. Apvrille & P. De Saqui-Sannes (2013): *Requirements Analysis*. *Embedded Systems: Analysis and Modeling with SysML, UML and AADL*.

[4] L. Apvrille, W. Muhammad, R. Ameur-Boulifa, S. Coudert & R. Pacalet (2006): *A UML-Based Environment for System Design Space Exploration*. In: *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on*, pp. 1272 –1275, doi:10.1109/ICECS.2006.379694.

[5] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò & L. Vigneron (2005): *The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications*. In: *CAV*, pp. 281–285.

[6] F. Assolini (2012): *The Tale of One Thousand and One DSL Modems, Kaspersky Lab*. Available at `http://www.securelist.com/en/blog/208193852/The_tale_of_one_thousand_and_one_DSL_modems`.

[7] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone & A. Sangiovanni-Vincentelli (2003): *Metropolis: An Integrated Electronic System Design Environment*. *Computer* 36(4), pp. 45–52.

[8] J. Bengtsson & W. Yi. (2004): *Timed Automata: Semantics, Algorithms and Tools*. In: *Lecture Notes on Concurrency and Petri Nets*, W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, pp. 87–124.

[9] B.Kordy, P. Kord, S. Mauw & P. Schweitzer (2013): *ADTool: Security Analysis with Attack-Defense Trees*. In: *QEST'13, LNCS vol. 8054, p. 173-176, Springer.*

[10] B. Blanchet (2009): *Automatic Verification of Correspondences for Security Protocols*. *Journal of Computer Security* 17(4), pp. 363–434.

[11] M. Drouineaud, M. Bortin, P. Torrini & K. Sohr: *A First Step Towards Formal Verification of Security Policy Properties for RBAC*. In: *QSIC'04*, Washington, DC, USA, pp. 60–67.

[12] D. P. Eames & J. D. Moffett (1999): *The Integration of Safety and Security Requirements*. In: *SAFECOMP*, pp. 468–480.

[13] S. Esser (2011): *iOS Kernel Exploitation*. In: *BlackHat 2011.*

[14] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle & B. Weyl: *Security Requirements for Automotive On-Board Networks*. In: *ITST 2009, Lille, France.*

[15] T. Hoppe, S. Kiltz & J. Dittmann (2011): *Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures*. *Rel. Eng. & Sys. Safety* 96(1), pp. 11–25. Available at `http://dx.doi.org/10.1016/j.ress.2010.06.026`.

[16] A. Huang (2002): *Keeping Secrets in Hardware: the Microsoft XBox Case Study, AI Memo 2002-008, Massachusetts Institute of Technology, Artificial Intelligence Laboratory*. Technical Report.

[17] M. S. Idrees (2012): *A Requirements Engineering Driven Approach to Security Architecture Design for Distributed Embedded Systems*. Ph.D. thesis, Telecom ParisTech.

[18] J. Jürjens (2002): *Using UMLsec and Goal-Trees for Secure Systems Development*. In G. B. Lamont, H. Haddad, G. Papadopoulos & B. Panda, editors: *Proceedings of the 2002 Symposium of Applied Computing (SAC)*, ACM, Madrid, Spain, pp. 1026–1031. Proceedings of the 2002 ACM Symposium of Applied Computing.

[19] J. Jürjens (2007): *Developing Secure Embedded Systems: Pitfalls and How to Avoid Them*. In: *29th International Conference on Software Engineering (ICSE 2007)*, ACM, pp. 182–183.

[20] E. Kelling, M. Friedewald, T. Leimbach, M. Menzel, P. Säger, H. Seudié & B. Weyl (2009): *Specification and Evaluation of e-Security Relevant Use cases*. Technical Report Deliverable D2.1, EVITA Project.

[21] D. Knorreck (2011): *UML-based Design Space Exploration, Fast Simulation and Static Analysis*. In: *Ph.D. of Ecole doctorale informatique, télécommunications et électronique of Paris.*

[22] D. Knorreck, L. Apvrille & P. De Saqui-Sannes (2011): *TEPE: A SysML Language for Time-Constrained Property Modeling and Formal Verification*. *ACM SIGSOFT Software Engineering Notes* 36(1), pp. 1–8.

[23] A. Maña & G. Pujol (2008): *Towards Formal Specification of Abstract Security Properties*. In: *The Third International Conference on Availability, Reliability and Security*, 0-7695-3102-4/08, IEEE.

[24] A. Nhlabatsi, B. Nuseibeh & Y. Yu (2010): *Security Requirements Engineering for Evolving Software Systems: a survey*. Technical Report 1, The Open University.

[25] B. Nuseibeh (2001): *Weaving Together Requirements and Architectures*. *IEEE Computer* 34(3), pp. 115–117.

[26] G. Pedroza (2013): *Assisting the Design of Secured Applications for Mobile Vehicles*. In: *Ph.D. of Ecole doctorale informatique, télécommunications et électronique of Paris.*

[27] G. Pedroza, M. S. Idrees, L. Apvrille & Y. Roudier (2011): *A Formal Methodology Applied to Secure Over-The-Air Automotive Applications*. In: *VTC-Fall2011, IEEE 74th Vehicular Technology Conference, 5-8 September 2011, San Francisco, USA*, San Francisco, USA.

[28] L. Pietre-Cambacedes & M. Bouissou (2013): *Cross-fertilization between safety and security engineering*. *Rel. Eng. & Sys. Safety* 110, pp. 110–126.

[29] C. Raspotnig & A. L. Opdahl (2013): *Comparing risk identification techniques for safety and security requirements*. *Journal of Systems and Software* 86(4), pp. 1124–1151.

[30] Y. Roudier, M. S. Idrees & L. Apvrille (2013): *Towards the Model-Driven Engineering of Security Requirements for Embedded Systems*. In: *proceedings of MoDRE'13, Rio de Janeiro, Brazil.*

[31] A. Ruddle & et al (2009): *Security Requirements for Automotive On-board Networks Based on Dark-side Scenarios*. Technical Report Deliverable D2.3, EVITA Project.

[32] B. Schneier (1999): *Attack Trees: Modeling Security Threats*.

[33] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille & D. Scheuermann (2011): *C2X Communication: Securing the Last Meter*. In: *The 4th IEEE International Symposium on Wireless Vehicular Communications: WIVEC2011*, San Francisco, USA.

[34] H. Teso (2013): *Aircraft Hacking*. In: *HITB Security Conference*, Amsterdam, The Netherlands.

[35] M. J. Toussaint (1993): *A New Method for Analyzing the Security of Cryptographic Protocols*. In: *Journal on Selected Areas in Communications*, 11, No. 5, IEEE.

[36] A. Van Lamsweerde (2007): *Engineering Requirements for System Reliability and Security*. *Software System Reliability and Security* 9, pp. 196–238.

Towards automating the construction & maintenance of attack trees: a feasibility study

Stéphane Paul
Thales Research & Technology
Palaiseau, France
stephane.paul@thalesgroup.com

Security risk management can be applied on well-defined or existing systems; in this case, the objective is to identify existing vulnerabilities, assess the risks and provide for the adequate countermeasures. Security risk management can also be applied very early in the system's development life-cycle, when its architecture is still poorly defined; in this case, the objective is to positively influence the design work so as to produce a secure architecture from the start. The latter work is made difficult by the uncertainties on the architecture and the multiple round-trips required to keep the risk assessment study and the system architecture aligned. This is particularly true for very large projects running over many years. This paper addresses the issues raised by those risk assessment studies performed early in the system's development life-cycle. Based on industrial experience, it asserts that attack trees can help solve the human cognitive scalability issue related to securing those large, continuously-changing system-designs. However, big attack trees are difficult to build, and even more difficult to maintain. This paper therefore proposes a systematic approach to automate the construction and maintenance of such big attack trees, based on the system's operational and logical architectures, the system's traditional risk assessment study and a security knowledge database.

1 Introduction

Large industries perform security risk assessments for large new systems whose development life-cycles span over many years. The security risk assessment work is to begin very early in the development life-cycle of the systems so as to positively influence the design towards secure architectures; the work is difficult because it has to be run on a yet poorly defined system architecture and because the risk assessment has to consider the complete system, i.e. an aggregation of premises, equipment, people and procedures. The amount of work justifies that multiple security experts are involved on the project, possibly from multiple sites and background. In this context, traditional risk assessment approaches, usually based on ISO-31000 [3] and ISO-27001 [4], reach their limit, in particular with respect to human cognitive scalability. On the European Galileo programme, the use of graphical attack trees has been shown to provide a positive contribution to traditional risk assessment approaches. However, the construction and maintenance of attack trees are tedious and error-prone tasks. Moreover, in a multi-user context, construction approaches differ from one security expert to another, which renders the reading and maintenance of attack trees even more complex for third-parties.

This paper reports on a feasibility study that was run to assess if it is possible to automate the construction & maintenance of attack trees using yet poorly defined architectural artefacts. This paper proposes a preliminary layer-per-layer systematic approach to generate skeletons of attack trees based on information coming from the system's architecture, as classically established using an architecture framework (AF), and information coming from the system's security risk assessment study, as traditionally run using a given risk assessment method and its related security knowledge base, e.g. EBIOS [1]. The support of an industrially used AF was a very important constraint imposed on this feasibility study; indeed,

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

we wanted to assess how much of an attack tree could be generated without additional work from system architects and/or designers. This feasibility study was run using empirical data from the Galileo risk assessment programme (cf. chapter 2 below), but obvious confidentiality reasons do not permit us to publish such data. Thus, this paper details the steps of the approach, and illustrates it on a running example from the automotive domain, which is public and easily understandable by all.

2 Scientific and empirical baseline to defining the approach

Galileo is Europe's own global navigation satellite system, providing a highly accurate, guaranteed global positioning service under civilian control. It is inter-operable with GPS and GLONASS, i.e. the US and Russian global satellite navigation systems. It corresponds to a system based on 30 satellites, 2 main control centres and 25 unmanned world-wide sites. The system is developed by the European Space Agency (ESA) for the European Commission (EC). To satisfy the security objectives identified in the frame of the Galileo programme, it was requested to define a risk management process. This needed to be an evolving process, taking into account the global life cycle of the system, from its specification to its operation. The main objective of the process to be defined was to provide the security accreditation authority assurance that the Galileo system is secure enough for authorising: (i) the launches of the satellites; (ii) the deployment of the ground infrastructure; (iii) the operation of the Galileo services.

Considering the above, the risk management process was defined through a series of brainstorming sessions, involving 10 experts from EC, ESA and industry, over a period of approximately 6 months. Techniques used in cyber-security and safety were analysed with respect to their applicability at system-level, whereby the system comprises equipment, people and procedures. Initial risk management process proposals were consolidated by assessing their direct application on the Galileo programme itself. The process was finally approved by the 27 Member States, in Sept. 2011, just before the first satellite launch. It is now proposed as the reference risk management process for other major EC programmes, e.g. EGNOS.

The graphical representation of risks on attack trees permitted the security accreditation authority to: (i) gain an intuitive approach of the risk, (ii) associate each risk to the Galileo System Design; and (iii) better perceive the impact of risk treatment on the system architecture. The success of the process defined for the Galileo programme has conducted Thales to generalise the risk management process for all IT systems under its responsibility. The approach proposed in this document is in a large part based on the empirical background gained through the Galileo programme. It also rests upon an important state of the art analysis [5] [11]. It will be further consolidated on other commercial risk assessment studies.

3 The running example

The running example is based on the *Loss of Integrity the Manual Breaking capability in a standard modern Car used as Taxi*. The running example has been modelled using the Thales Melody AF. It is a simplistic model; the objective here is not to build a realistic car model, but to include the key modelling artefacts that can be used to generate an attack tree.

The Thales Melody AF supports architecting using four abstraction levels, as illustrated in Figure 1. Under our assumption that the security risk assessment is being performed early in the system's development life-cycle, the `Physical` abstraction level of Melody has not been used in our approach.

The Thales Melody Architecture Frameworks is UMLish. It is our believe that most state-of-the-art and commercially available AFs would provide similar capabilities to support the proposed approach, so

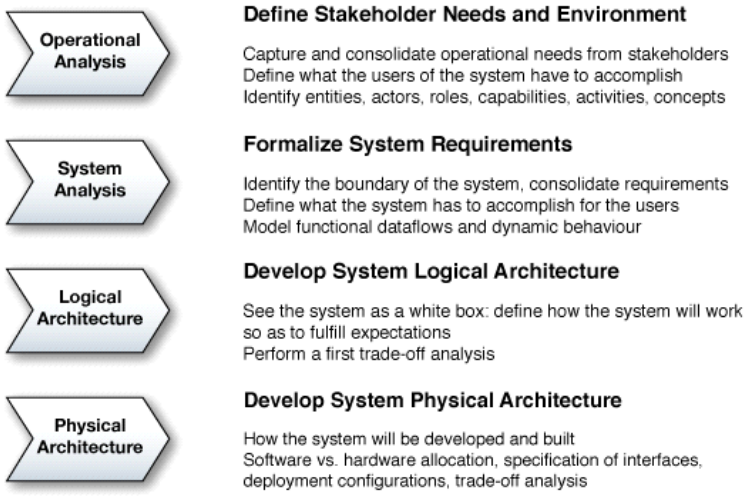

Figure 1: The four abstraction levels of the Thales Melody framework

alternative AFs could have been used, but a detailed assessment has not (yet) been performed.

4 Attack tree construction overview

Our approach uses the dominant type of attack tree model, i.e. the Boolean-logical tree based approach in which the top or root node of the tree represents a defender's feared event. Given a feared event, our proposal is to systematically structure the attack tree in layers according to the following elements:

- system states and modes, in which the occurrence of the feared event makes sense;
- supporting asset types (e.g. hardware, software) that could potentially be attacked;
- attack entry points (i.e. supporting asset interfaces) for each supporting asset type;
- threats that can be exercised on the attack entry points;
- threat sources that can exercise the threats.

The two high-level principles governing the tree construction are the following:

- the feared events are defined at strategic level: they are driven by operational considerations,
- logical `AND` gate decompositions should be located as low as possible in the tree.

The first principle originates from the customer and enforces a true risk-based approach (by opposition to technical security). Moreover, when the feared event at the root of an attack tree is driven by operational considerations, it is simple to assign an individual stakeholder to the attack tree, with which the security expert will be able to discuss the relevance and completeness of the attack tree decomposition.

The second principle is driven by attack tree exploitation and ergonomic considerations. Conjunctive Boolean gates create dependencies between tree branches. Locating `AND` gates as low as possible in the tree limits the span of dependent branches and increases readability of the attack trees. In this set-up, `AND` gates are typically used to capture:

- pre-conditions to attack enacting, for example, knowledge about the supporting asset, or, more complex, a change in state and mode;
- conditions to make succeed the attack, in particular with respect to system redundancy;
- post-conditions, typically to allow for the repudiation of the attack.

5 Step-by-step description of the approach

5.1 Step 1: Creation of the attack tree root

Feared events are classically captured textually in the security risk assessment study. For our running example, let us suppose that the feared event is defined as the *Loss of Integrity of the Manual Braking operational process on the Car*. Thus, it is easy to extract the feared event from the security risk assessment study to create the attack tree root. The technical challenge here is to map this informal statement with artefacts of the system's architecture and security knowledge base.

From the system's operational architecture (cf. Figure 2) it is possible to recognise the `Car` as being an Operational Entity (OE). The `Manual Braking` operational process is pictured as a sequence of operational activities (OAs); the first activity (i.e. `Dynamic Sensing`) and the last activity (i.e. `Brake`) of the operational process are surrounded by thick borders; the interactions between the operational activities are pictured by thick arrows; other (thin) arrows represent interactions that are not part of the `Manual Braking` operational process.

From our security knowledge base, it is possible to recognise the term `Integrity` as being a standard security criterion; `Confidentiality` and `Availability` would likewise be eligible.

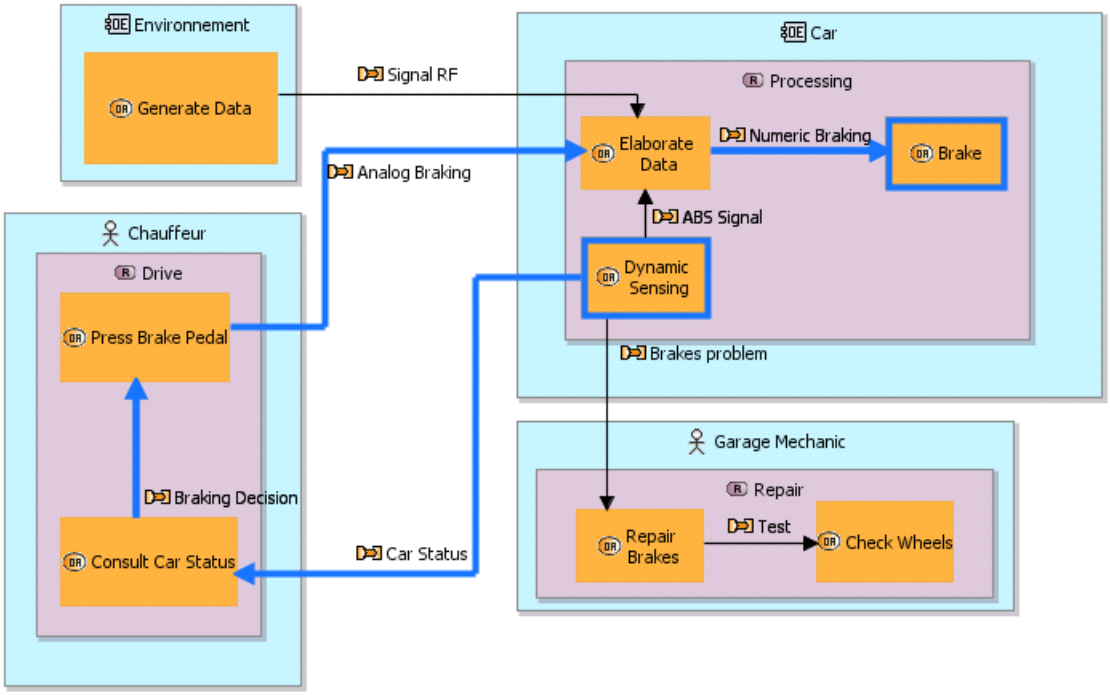

Figure 2: Running example - the manual braking operational process

Our proposal is not to perform automatic recognition on informally stated feared events, but rather to impose a strict grammar to the description of feared events in the security risk assessment study, typically:

Loss of [*security criterion*] of the [*primary asset*] on the [*operational entity*].

To capture the security criterion, the security expert would select an option from a drop down menu, or similar mechanism, in line with the security knowledge base selected for the study. To capture the operational entity (i.e. the `Car`), the security expert would select the artefact directly from the system's operational architecture by browsing through the list of defined operational entities. Such an approach has already been successfully experimented in a security risk assessment context [10]. In the above construct, the `Manual Braking` operational process of the AF is not referenced directly. Instead, we have used the concept of *primary asset*, as defined in the French EBIOS risk assessment method, i.e. something that is of value [1]. It is assumed that the primary asset has been previously mapped to the operational process artefact in the system's operational architecture (i.e., the `Manual Braking` operational process, in a similar manner as for the operational entity above [10].

The severity of the feared event originates from the risk assessment study. In our running example, it is assumed to be `Critical`.

5.2 Step 2: Structuring the tree according to states and modes

The second step in constructing the attack tree is to take into consideration the system's states and modes in which the realisation of the considered feared event makes sense. The Melody AF allows for a *state machine X operational activities* matrix, which defines in which states and modes the operational processes can be run. From this matrix, the attack tree can be generated as follows:

- a first decomposition of the feared event is made with all the states in which the operational process can run, connected with an `OR` gate;
- a decomposition of the state nodes can then be made with all the modes in which the operational process can run, connected with an `OR` gate to the corresponding state node in the tree.

For our running example, let us consider a simple state machine, as illustrated in Figure 3. Let us also make the reasonable assumption that the `Manual Braking` operational process can be run only in the `Operating` state and in the `Engine Running` mode of the `Car`. Thus, in the attack tree, there is need for only one branch corresponding to the `Operating` state, and there is need for only one branch corresponding to the `Engine Running` mode. The resulting tree is shown in Figure 4.

It is noteworthy that the labelling of the attack tree nodes can be automatically and dynamically generated based on the system's operational architecture. Dynamicity stems from the traceability links that are established between the tree nodes and the states and modes in the operational architecture. Thus, if the name of a state or mode changes, the label of the corresponding node in the attack tree can be automatically and immediately updated. Likewise, if a state or mode is deleted, a warning can be attached to the corresponding subtree, calling for specific attention by the security expert. The running example illustrated here is obviously simplistic, but it is sufficient to capture the general idea that an attack depends on the states and modes of the targeted system.

It is worthwhile recalling here that elaborating the attack tree with states and modes is not just a tree structuring feature. States and modes carry semantics about system behaviour, and therefore also about what attacks are possible at a given time. In step 6 (see below), we introduce attack preconditions; some of these conditions can be a change of system state or mode, which inherently transforms the attack tree into a Directed Acyclic Graph (DAG) or creates duplicate sub-trees.

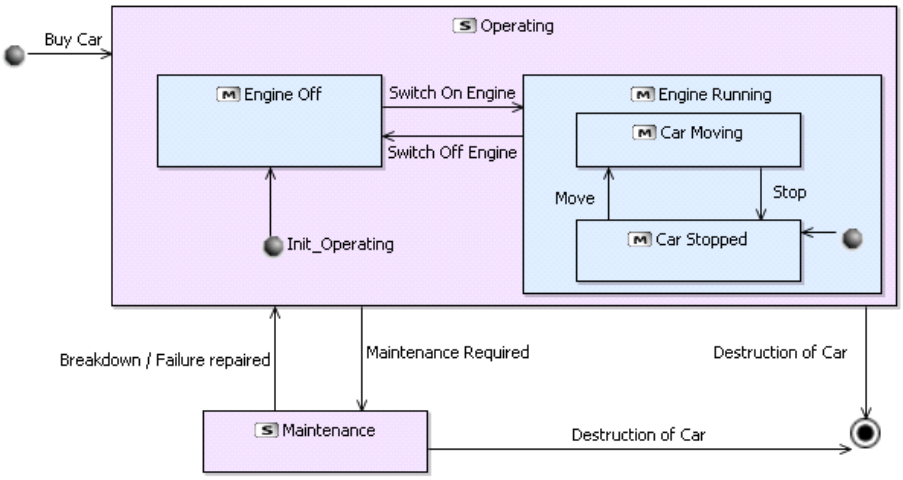

Figure 3: Running example - the manual braking operational process

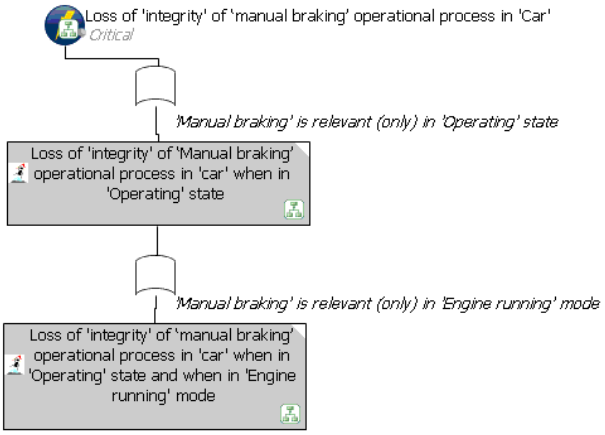

Figure 4: Automatically constructed attack tree based on states and modes

5.3 Step 3: Structuring the tree according to supporting asset types

Security knowledge bases often organise supporting assets in types in order to define categories of vulnerabilities with their corresponding threats, e.g. paper documents can burn, hardware can be stolen, networks can be saturated, people can be influenced. In this 3rd step of the attack tree construction, we make use of this supporting asset typology in order to map the attack tree structure with the knowledge base. The goal here is to increase the readability of the attack tree and assure its completeness with respect to best practices as captured in the security knowledge bases. However, this structuring is optional and can be skipped if felt as unnecessary.

For our running example, let us suppose the existence of four supporting asset types: `Hardware`, `Software`, `Networks` and `Organisations`. At this very early stage in the architectural design, `Network` must not be understood as routers, servers, switches, and the like, but rather as functional and / or component exchanges, including social networks. Focus is more on the data that is exchanged in support of the operational processes, than on the equipment that supports the data exchanges.

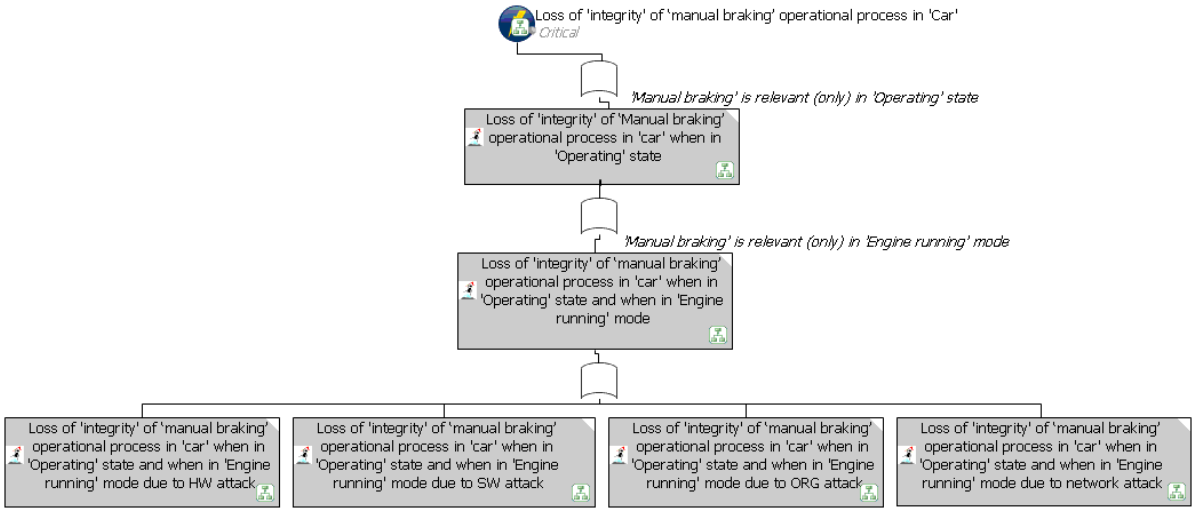

Figure 5: Automatically constructed attack tree based on supporting asset types

The resulting tree is shown in Figure 5. As in the previous step, the labelling of the attack tree nodes can be automatically generated based on the content of the security knowledge base.

5.4 Step 4: Structuring the tree according to attack entry points

During the fourth step, the objective is to capture the supporting assets that can be targeted as attack entry points by the attacker. This exercise requires identifying all the possible attack entry points defined in the logical architecture, supported by the traceability links between the artefacts of the operational architecture and their corresponding elements in the logical architecture (cf. Figure 1).

Figure 6 provides a simplified logical architecture model for our running example. In this model, a `Manual Braking` functional chain has been defined that corresponds to the `Manual Braking` operational process defined in the operational architecture. The Thales Melody AF allows one to trace artefacts between abstraction layers, as illustrated for our running example in Figure 7.

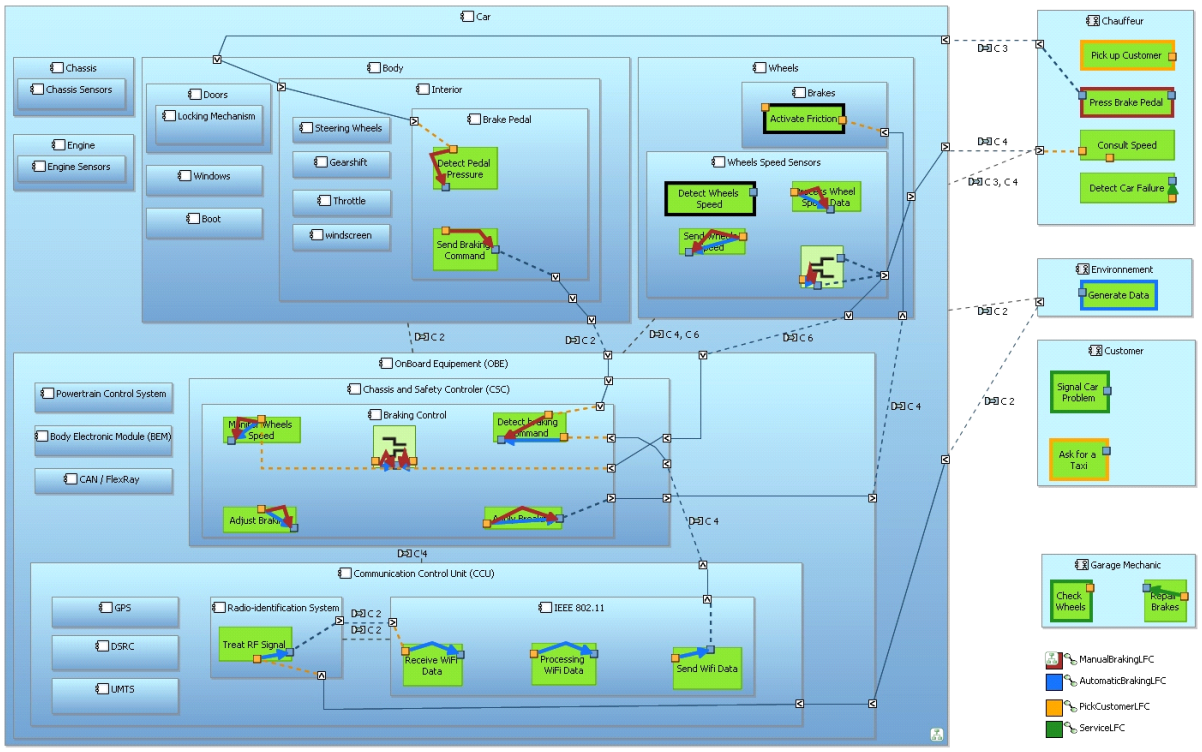

Figure 6: Running example - simplified logical architecture model

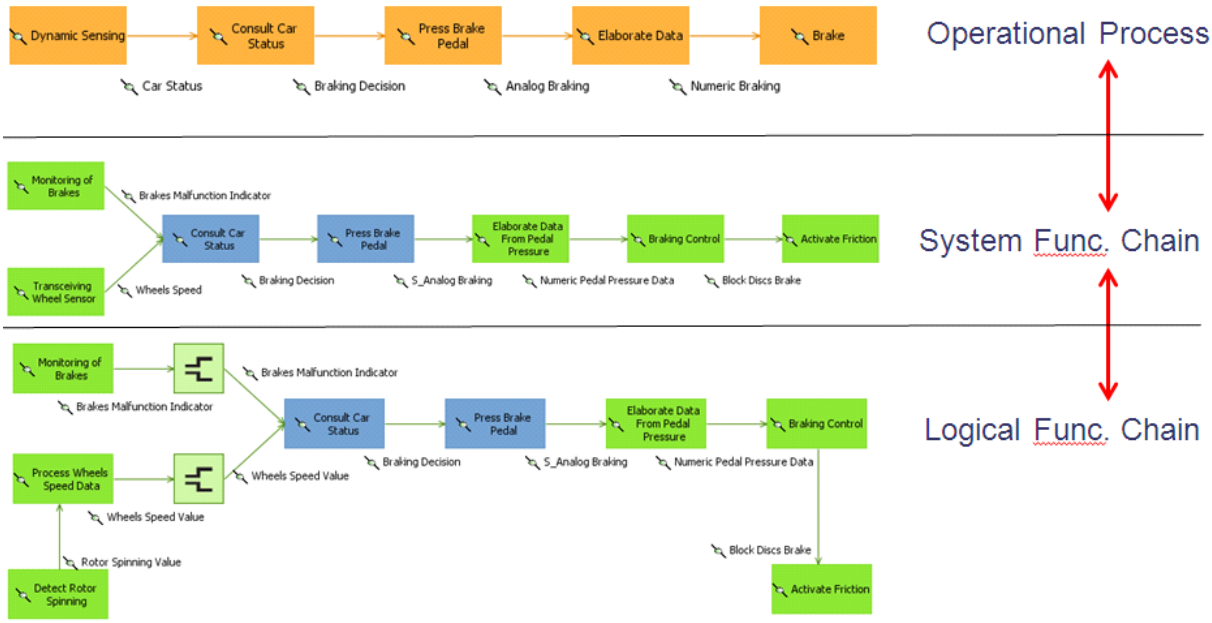

Figure 7: Running example - traceability between the operational process and the logical functional chain

In the Melody AF, and presumably in most logical architecture frameworks, the logical components are not assigned tags specifying their type. However, this work is traditionally done in security risk assessment studies, as illustrated for our running example in Figure 8. In this figure it can be seen that:

- the `Brake Pedal` and `Brakes` (meaning here `Break Pads`) are hardware components,
- the `Customer`, `Chauffeur` and `Garage Mechanic` are people, and
- the `Braking Control` and `Wheel Speed Sensors` are complex systems, including hardware, software and / or network sub-components.

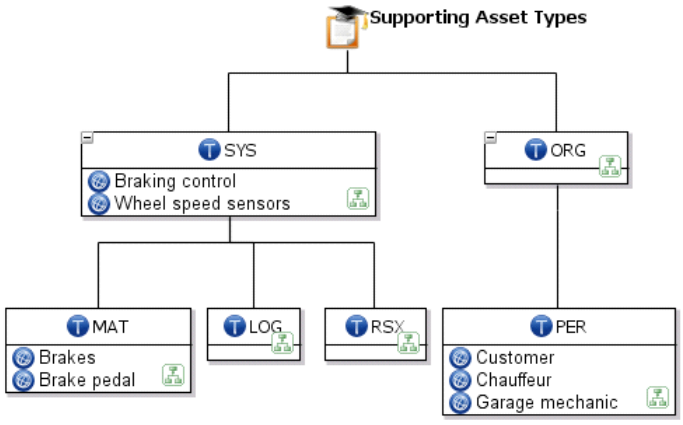

Figure 8: Running example - tagging of supporting assets in the security risk assessment study

Based on all the aforementioned information, it is now possible to continue the decomposition of the attack tree: for all leaf nodes of the attack tree corresponding to `Hardware`, `Software` and `Organisations` add the corresponding identified attack entry points, connected with `OR` gates, if they are involved in the logical functional chain corresponding to the operational process referenced in the feared event at the tree root.

For `Network`-related nodes, the construct is a bit more complex. The objective here is to identify all the (external) logical component interfaces that could be used as attack entry points in threatening the functional chain. The search of these relevant logical component interfaces is highly dependant on the selected Architecture Framework, and will not be detailed here. Assuming they are correctly identified, it is possible to complete the attack tree decomposition: add all the identified attack entry points related to interfaces under the `Network` leaf node of the tree, connected with an `OR` gate.

In our running example, the two hardware supporting assets can be added below the hardware attack node, connected with an `OR` gate. For people, only the `Chauffeur` needs to be added, because the other two persons are not involved in the logical functional chain. For elements typed as SYS (cf. Figure 8), new nodes need to be created both under the `Hardware` and `Software` attack nodes; the construction process is similar to the one described above, but it should be noted that SYS elements lead either to the duplication of nodes in the tree, or to the creation of a directed acyclic graph (DAG) if the tooling supports such a construct. The management of the `Network` components of SYS elements is explained below. In our simple running example, we have defined two external interfaces:

- the `Dashboard`, which allows for the display of the car's speed and also provides a malfunction indicator;

- the `Brake Pedal`, which allows the driver to apply an analogic braking force that somehow translates to a braking force on the brake pads.

Only the `Dashboard` interface is concerned by the two logical components tagged as `SYS`. Thus only the `Dashboard` interface should be added in the attack tree under the `Network` attack node. The complete resulting attack tree (in fact a DAG), at the end of this step, is illustrated in Figure 9.

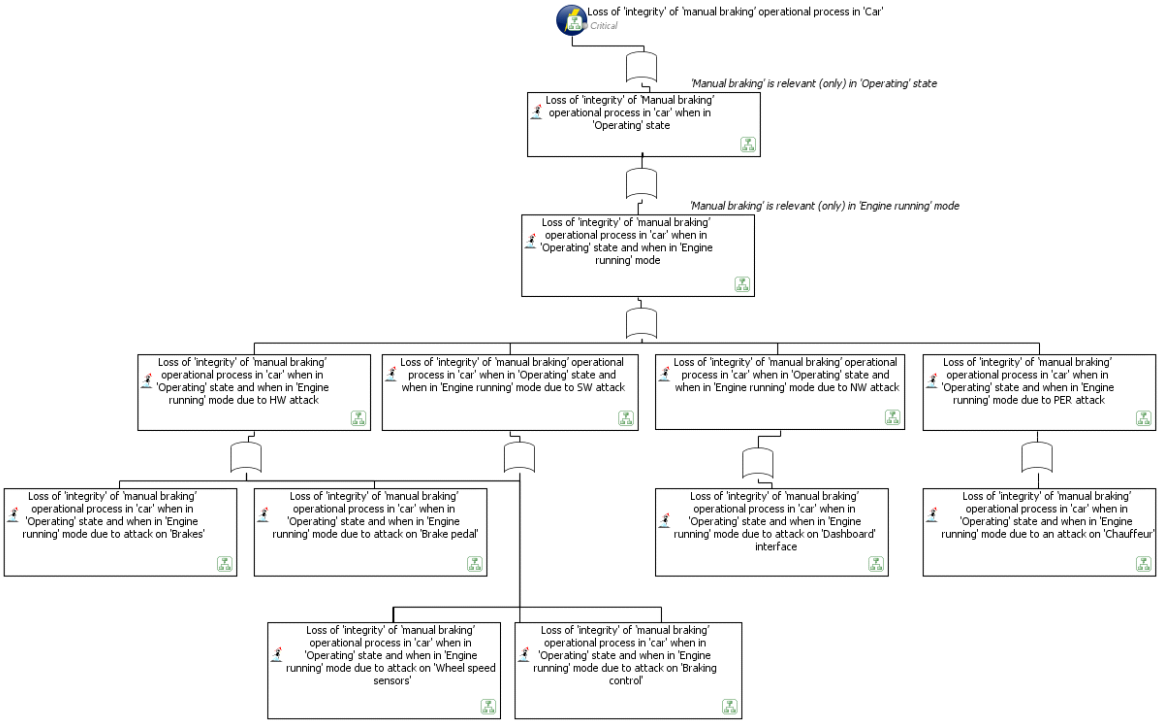

Figure 9: Automatically constructed attack tree based on attack entry points

5.5 Step 5: Structuring the tree according to applicable threats

In this fifth step, we again use the security knowledge base to decompose each supporting asset leaf node of the attack tree with all realistic threats that can apply, connected with an `OR` gate. We assume that the security knowledge base provides a list of threats characterised by at least the targeted supporting asset type and the concerned security criteria, as illustrated in Table 1. The proposed algorithm systematically scans the security knowledge base selected for the study; it therefore ensures the completeness of the threat study with respect to that database. As in the previous steps, the labelling of the attack tree nodes can be automatically generated based on the content of the security knowledge base.

According to the EBIOS knowledge base [1], amongst the six threats targeting people, only the `Influence over a person` and `Overloading of the capacity of a person` threats are related to the `Integrity` security criterion. Thus, in our running example, the attack node on the `Chauffeur` can be decomposed with two attack sub-nodes. All the other attack entry points are treated similarly. Due to space limitations in this paper, the resulting tree is not shown.

Table 1: Examples of threat descriptions in the EBIOS knowledge base [1]

Threat	**Targeted supporting asset**	**Description**	**Concerned security criteria**	**Exploited vulnerabilities**	**Pre-requisites**
MAT-MOD	Hardware	Hardware modification	Availability, Integrity, Confidentiality	Elements can be added, retrieved or substituted; Elements can be deactivated	Knowledge of the existence and location of the hardware; Physical access to the hardware
RSX-USG	Network	Man-in-the-middle attack	Availability, Integrity	Flow content can be altered; Flow rules can be altered; Is the unique transmission resource	Knowledge of the existence and location of the canal; Physical or logical access to the canal

5.6 Step 6: Structuring the tree according to threat sources

The sixth and last step of our algorithm is the most complex. For this step, we make the reasonable assumption that the threat sources are explicitly defined in the risk assessment study. The complexity of this algorithm step stems from the necessity of adequately selecting the threat sources to be added in the attack tree, based on the fact that they can realistically enact the threats. Selecting unrealistic threat sources may lead the attack tree end-users to reject the attack tree because uselessly oversized; on the contrary, retaining too few threat sources may compromise the completeness of the study.

Each threat has prerequisites, in particular attack entry point access prerequisites that must be satisfied by a threat source for that threat source to be retained in the attack tree. For example, to be able to modify some hardware equipment, a threat source requires knowledge about the existence & location of the hardware, and needs physical access to that hardware equipment (cf. Table 1).

The system's logical architecture provides information on the existence of access possibilities, typically based on the existence of logical component ports. In our running example, access to the `Brake Pedal` is possible only through the `Body` and `Interior` of the `Car`.

The system's logical architecture also provides information on the expected uses of these access possibilities, typically based on functional chains. In our running example, Figure 6 shows that access to the `Brake Pedal` is performed as part of the `Manual Braking` functional chain.

Finally, the system's logical architecture provides information on the expected users of these access possibilities, typically based on operational actors some of which may be considered as threat sources. In our running example, Figure 6 shows that the `Chauffeur` is an operational entity that is expected to access the `Brake Pedal` as part of the `Manual Braking` functional chain. If the `Chauffeur` is expected to use the `Brake Pedal`, then it can be assumed that he has knowledge about the existence and location of the `Brake Pedal`; he may even have received training about it. Thus, if a threat source is part of the system's logical architecture (i.e. insider attacker), some attack entry point access preconditions can be checked.

However, all preconditions cannot be checked, even when the threat source is part of the system's logical architecture. For example, the `Taxi Customer` has access to the `Car Interior` but he is not expected to use the `Brake Pedal`, so, based on the logical architecture, no assumption can be made on

his knowledge of the existence and localisation of the Brake Pedal. Indeed, a system design expresses what a system is or can do, not what it is not or cannot do. This is a major limitation in using architectural artefacts for building attack trees. This limitation can be overcome by requesting additional expert input, but such additional inputs must be kept as low as possible to preserve the usability of the approach.

Due to paper length limitations, the conditions determining if a threat source should or should not be retained for insertion in the attack tree cannot be discussed herein. This paper however exposes the two main issues that render quite complex this step of the automation approach. It also provides the high-level algorithm and illustrates the resulting tree with the running example.

There are two main issues in selecting threat sources for insertion in the attack tree: (1) it is required to distinguish between physical and logical access pre-conditions; (2) access prerequisite satisfaction, for a known threat source, depends on the system's state and mode.

The proposed high-level algorithm to decide whether to retain or reject a threat source is the following:

- if the threat source is not represented as an actor in the architecture, then the threat source is retained because we lack knowledge about it: the security expert is requested to further manually develop or close this branch of the attack tree;
- else if the threat source has the required access (physical and/or logical, as required for the attack) to the attack entry point in the considered system's state & mode then the threat source is retained, for an intentional or an accidental attack; the security expert is requested to further manually develop or close this branch of the attack tree;
- else if the threat source is malevolent, the threat source is retained, even though there is a doubt about its access capabilities; the security expert is requested to further manually develop or close this branch of the attack tree;
- else (i.e. known non-malevolent threat source with no known access to the supporting asset), the threat source is rejected.

If a threat source is retained to be added in the attack tree for a given attack, then an AND gate is added with:

- one or more leaf nodes representing the attack preconditions, as provided in the security knowledge base (cf. Table 1);
- the attack itself; the security expert is requested to further manually develop this attack node;
- optionally, attack post-conditions, typically to ensure attack repudiation.

Amongst the most interesting preconditions are the ones involving a change of state and mode, which can be far in the past, or immediately before the attack. In our running example, activating malicious code in the Braking Control system when the Car is moving requires having injected that malicious code when the Car was in the Maintenance state. Here, the precondition is represented by a very complex subtree, but an online change of state and mode can be even more complex. Indeed, the attack is then enacted in the new state and mode, possibly with an impact on the severity, or even the relevance, of the feared event.

This algorithm step is the first in which we use the AND logical gate, in accordance with our aim to introduce conjunctions as low as possible in the attack tree.

Figure 10 shows the result of the application of the algorithm on approximately half the attack nodes. The attack tree is not meant to be legible, but is shown here to give an idea of the complexity of the generated trees. In the Galileo risk assessment programme, some attack trees are known to stretch over

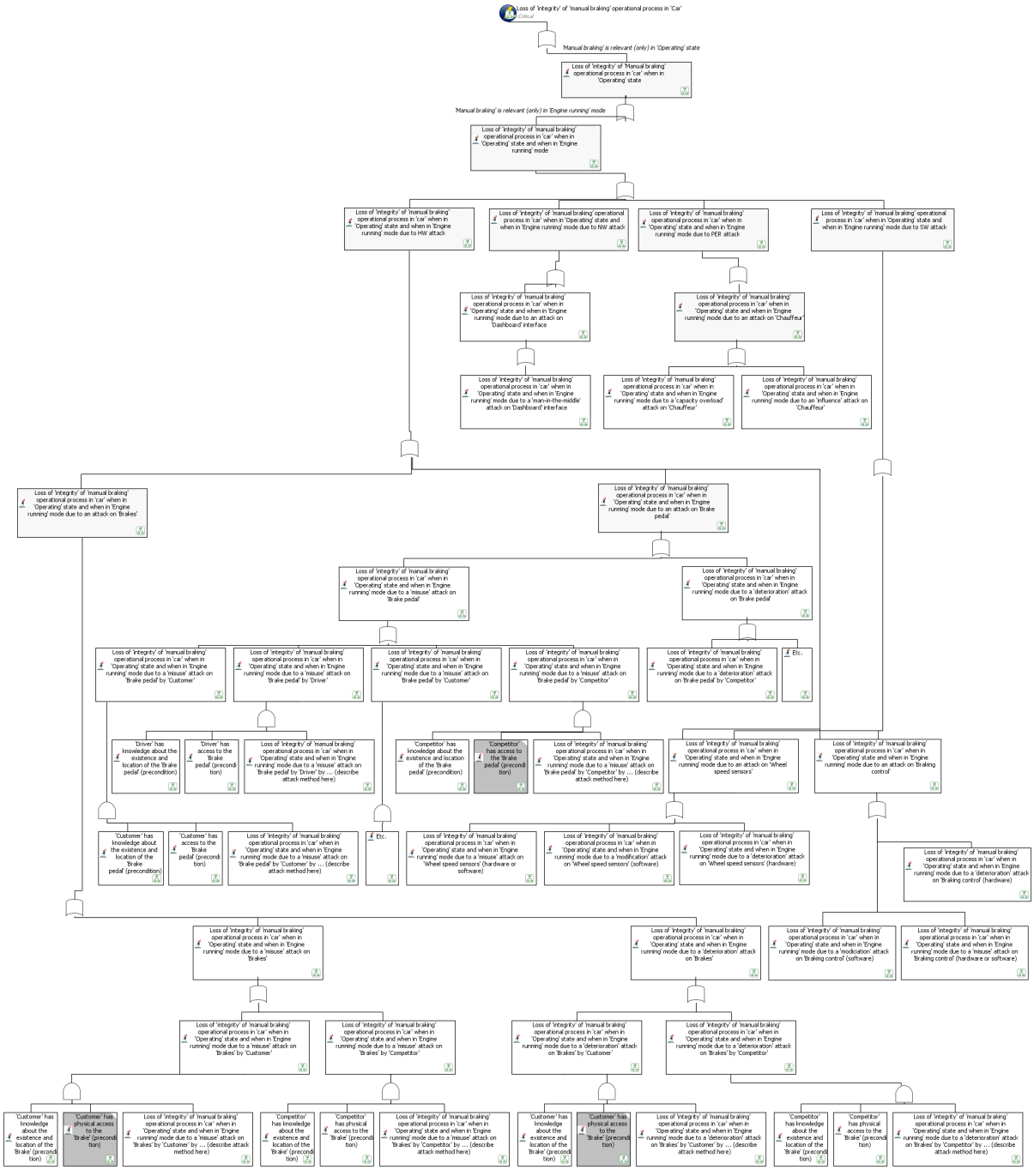

Figure 10: Automatically (partially) constructed tree based on retained threat sources

40 A4 pages. In our simple running example, the attack tree size is expected to more than double when the decomposition process is completed.

6 Related work

There are few commercial products related to attack trees. The two most significant ones are SecurITree [9] and AttackTree+ [7]. None of them proposes support for the automated construction of attack trees.

The scientific community is very active on the construction of attack graphs [8, 2]. However, attack graphs essentially focuses on attacker attempts to penetrate well-defined computer networks, rather than addressing socio-technical systems, especially when poorly or partially defined; and naturally, the complexity issues addressed by this community essentially relates to scaling to very large networks, rather than to usability, i.e. trying to construct an attack tree skeleton that can be easily reworked and maintained by human security experts.

Some research papers do address socio-technical systems. For example, [6] and [12] propose formal approaches to generate attack trees, respectively based on system goals, and security policies. These studies can be seen as upstream complements, but they also require specific frameworks. Our approach attempts to base most of its attack tree extraction on an industrially used framework.

7 Conclusion and way forward

As can be seen from Figure 10, significant *draft* trees can be automatically generated, even with a simple case-study. Beyond saving security expert time to build the attack tree, a systematic approach is enforced, which ensures the completeness of the threat and vulnerability analysis. Moreover, the top-level structure of the trees is normalised throughout the study, thus easing the readability and understanding of the trees by third-parties; only the lower parts of the tree are left for manual completion. Last but not least, traceability to architecture artefacts is set-up at no additional cost, thus offering support for impact analysis when the system architecture evolves.

Our approach is foreseen for use only for large new systems, and not to support change in existing systems. The main reason for that is that one starts the design of a large new system with a white sheet; in this setting, the approach allows for the rapid construction of some core attack trees, and it eases the impact management as the system's design evolves. When system delivery is concluded, the commercial contract(s) for system provisioning and system risk assessment usually terminate(s). A new commercial contract may now be enacted to support Security Information and Event Management (SIEM) at operation-time, possibly with a different industry. There are multiple approaches to SIEM, but most are based on attack graphs and / or Complex Event Processing (CEP) techniques, rather than on attack trees. Reuse of the knowledge captured in attack trees to feed the SIEM would definitively be useful if the same industry manages to win both the design-time risk assessment study and the SIEM contract, but we have not (yet) investigated this research path.

To support the automatic attack tree generation, inputs are required from: (i) an architecture framework, in particular data from the operational architecture and the logical architecture; (ii) a risk assessment tool, in particular in terms of feared events, primary assets, supporting asset types, threat sources, etc.; (iii) a security knowledge base, in particular in terms of supporting asset types and threat caracterisation (cf. Figure 11).

Beyond the positive points stated above, this feasibility study has also highlighted some issues; further research has been shown to be required on a number of topics, in particular:

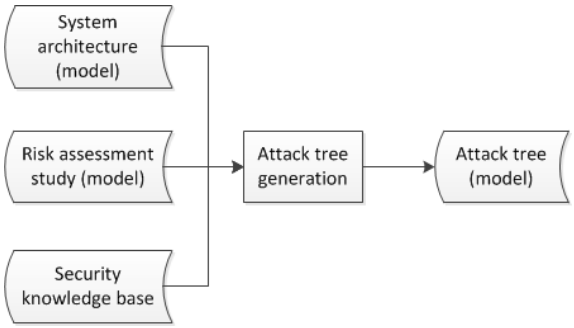

Figure 11: Inputs and outputs of the attack tree generation approach

- the current work was focused essentially on operational processes and / or logical functional chains; further research is required to cope with attacks on data and communications between logical components;
- the proposed algorithm has made the assumption that ports on logical component were tagged to specify if they provided logical versus physical access; this approach needs to be further validated because it requires additional work compared to current industrial engineering practices;
- the proposed approach extensively analyses the logical architecture in search of the relevant attack entry points, rather than using the supporting assets, as defined the EBIOS [1] risk assessment study; the consistency between attack entry points and supporting assets must be ensured;
- the high-level structuring of the attack tree is based on states and modes; it has been implicitly assumed in this feasibility study that there were only two levels (i.e. a state-level and a mode-level which refines the states) in the system's architecture; however, it can be supposed that very complex systems may use sub-states and / or sub-modes; further research is required to assess the importance of considering more than two level state machines;
- the approach fails to capture that some of the attacks may be led during the systems development life-cycle itself, e.g. theft of detailed design documents during the transmission of those documents between the development teams; to cover this type of attack, we may need to define a development life-cycle (e.g. specification, coding, integration) by opposition to a operation-time life-cycle, or, at minima, include in the attack tree an additional branch for attacks during the development phase, to be manually completed by the security expert; for certain programmes, it could also be interesting to include in the tree, attacks that could occur during system disposal;
- the feasibility study has been run using the Thales Melody architecture framework; synthesising and generalising the required inputs from the architecture framework is required, in particular to assess connection possibilities to other, possibly commercial, architecture frameworks; formalising the algorithm would then be possible, for example using the Object Constraint Language (OCL) if the architecture is expressed in the Unified Modeling Language (UML);
- logical `AND` gates have been introduced in the tree only to cope with attack pre- and post-conditions; further work is required to deal with logical redundancy in operational and / or functional chains;
- to ease the readability and understanding of the automatically generated tree, significantly long node labels have been used to precisely define the attack and its enactment conditions; since many attack tree tools, whether academic or commercial, do not support long node labels or restrict node

labels to a unique line, further research is required to assess the trade-off between readability and scalability; tree layout may also be a constraint to consider, as opening a poorly-formatted large automatically-generated attack tree may be perturbing for the end-user;

- attack tree node generation can sometimes be disputable; in those cases, we have always favoured node generation, so as to ensure the completeness of the analysis, making up for possibly unrealistic node generation by using node annotations or colour codes to attract the security experts' attention; this approach, originating from expert judgement, needs to be further validated;
- the threat source selection algorithm is the most complex part of the overall approach, but still remains the least convincing; further research is required to simplify the approach.

8 Acknowledgment

The research leading to these results has received funding from the ITEA2 MERgE project. The help and support from engineers at Thales Communications & Security have been of particular value.

References

[1] Agence nationale de la sécurité des systèmes d'information (2010): *EBIOS 2010 – Expression of Needs and Identification of Security Objectives*. In French.

[2] H. Birkholz, S. Edelkamp, F. Junge & K Sohr (2010): *Efficient automated generation of attack trees from vulnerability databases*. In: *Working Notes for the 2010 AAAI Workshop on Intelligent Security (SecArt)*, pp. 47–55.

[3] International Organization for Standardization (2009): *ISO 31000 – Risk management – Principles and guidelines*.

[4] International Organization for Standardization / International Electrotechnical Commission (2005): *ISO/IEC 27001 – Information technology – Security techniques – Information security management systems – Requirements*.

[5] Barbara Kordy, Ludovic Pietre-Cambacedes & Patrick Schweitzer (2013): *DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees*. *CoRR* abs/1303.7397. Available at `http://arxiv.org/abs/1303.7397`.

[6] Axel Van Lamsweerde, Simon Brohez, Renaud De Landtsheer & David Janssens (2003): *From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering*. In: *Proc. of RHAS03*, pp. 49–56.

[7] Isograph Ldt.: *AttackTree+ for Windows, Version 1.0, Attack Tree Analysis*. Available at `http://www.isograph.com/software/attacktree/`.

[8] R. Lippmann & K. Ingols (2005): *An annotated review of past papers on attack graphs*. Technical Report ESC-TR-2005-054.

[9] Amenaza Technologies Ltd.: *SecurITree, Attack tree modelling*. Available at `http://www.amenaza.com/`.

[10] Stéphane Paul & Olivier Delande (2011): *Integrability of design modelling solution*. SecureChange FP7 project deliverable D4.4b.

[11] Stéphane Paul, Raphael Vignon-Davillier, Quentin Guil, Mickael Malka & André Leblond (2013): *Understanding attack trees in the context of security risk assessment studies: a state of the art*. Thales technical report. Industry-in-confidence.

[12] W. Pieters, T. Dimkov & D. Pavlovic (2013): *Security Policy Alignment: A Formal Approach*. *IEEE Systems Journal* 7(2), pp. 275–287.

Possibilistic Information Flow Control for Workflow Management Systems*

Thomas Bauereiss Dieter Hutter
German Research Center for Artificial Intelligence (DFKI)
Bibliothekstr. 1
D-28359 Bremen, Germany
thomas.bauereiss@dfki.de hutter@dfki.de

In workflows and business processes, there are often security requirements on both the data, i.e. confidentiality and integrity, and the process, e.g. separation of duty. Graphical notations exist for specifying both workflows and associated security requirements. We present an approach for formally verifying that a workflow satisfies such security requirements. For this purpose, we define the semantics of a workflow as a state-event system and formalise security properties in a trace-based way, i.e. on an abstract level without depending on details of enforcement mechanisms such as Role-Based Access Control (RBAC). This formal model then allows us to build upon well-known verification techniques for information flow control. We describe how a compositional verification methodology for possibilistic information flow can be adapted to verify that a specification of a distributed workflow management system satisfies security requirements on both data and processes.

1 Introduction

Computer-supported workflows and business process automation are widespread in enterprises and organisations. Workflow management systems support the enactment of such workflows by coordinating the work of human participants in the workflow for human activities as well as by automatically executing activities that can be mechanised. Graphical notations such as BPMN allow for the specification of workflows in an intuitive way. In addition to the control and data flows, there are typically various security requirements that need to be considered during the design, implementation and execution of workflows. A well-known security requirement on workflows is separation of duty for fraud prevention [7]. Confidentiality of data is another important security requirement, e.g. the confidentiality of medical data from non-medical personnel. These two can be seen as examples for different types of security requirements. On the one hand, there are security requirements on processes, i.e. constraints on the control flow and the authorisation of users, and on the other hand, there are security requirements on data, i.e. constraints on the flow of information. Several proposals to extend BPMN with graphical notations for both kinds of security requirements exist [6, 26, 33].

In this paper, we focus on the question of how the semantics of such a notation can be defined and how to use them to formally verify both types of security requirements. We do this on an abstract level without having to refer to details of enforcement mechanisms such as role-based access control (RBAC). For this purpose, we model the behaviour of a workflow as a set of traces of events, each representing a possible run of the workflow, and formalise our security requirements in a declarative way as properties of such trace sets. We map process requirements such as separation of duty to sets of allowed traces, corresponding to safety properties [3], whereas we map requirements on data to information flow properties,

*This research is supported by the Deutsche Forschungsgemeinschaft (DFG) under grant Hu737/5-1, which is part of the DFG priority programme 1496 "Reliably Secure Software Systems."

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

which have been extensively studied [27, 15, 36, 21, 9]. This allows us to verify the absence not only of direct information leaks via unauthorised access, but also of indirect information leaks via observing the behaviour of the system. For example, if the control flow depends on a confidential data item and an unauthorised user observes which path of the control flow has been taken, they might be able to deduce the confidential value of the data item.

The relation between possibilistic information flow and safety properties is not trivial due to the refinement paradox, i.e. enforcing a safety property by removing disallowed traces might introduce new information leaks [18]. We discuss this relation for the case of separation of duty, give sufficient conditions for the compatibility with information flow properties, and show that these conditions are satisfied in our example setting.

We build upon the MAKS framework for possibilistic information flow control [15], which is suitable for formulating and verifying information flow policies at the specification level, and in which many information flow properties from the literature can be expressed. We describe how a compositional verification methodology [13] can be applied to verify our system models, which has the advantage that we can split up the verification task into separate verification tasks of the individual activities that make up the overall workflow.

Essentially, our approach allows for the formal modelling of workflows and the verification of security requirements on data and processes at a high level of abstraction. We have verified our results using the interactive theorem prover Isabelle/HOL [24]. To improve practicality, future work will focus on refinement approaches of these specifications towards concrete implementations, while preserving as much as possible of the security properties established on the abstract level. The long-term goal of our work is to facilitate the step-wise development of secure workflow management systems, starting from an abstract specification, derived from a workflow diagram, for example, then performing a series of refinement steps and eventually arriving at a secure implementation.

The rest of this paper is structured as follows. In the following subsection, we present a running example of a workflow that we will use for illustration throughout this paper. Section 2 introduces the system model. In Section 3, we elaborate on modelling confidentiality and separation-of-duty requirements, respectively. In Section 4, we describe how existing techniques for compositional verification of information flow properties can be applied and adapted for our workflow systems. Section 5 discusses related work and Section 6 concludes the paper.

1.1 Example Scenario

As a running example, we use the workflow depicted in Figure 1, adapted from an industry use-case described in an (unpublished) paper by A. Brucker and I. Hang. It models a hiring process including interviews and medical examinations. The swimlanes represent two departments of the organisation running the hiring workflow, the Human Resources (HR) and the medical department. The placement of activities in the swimlanes indicates the responsible department, and thus the authorised employees. The input and output relations of activities are depicted as directed flows of documents between activities.

We use this workflow to illustrate security requirements with respect to both data and process. On the process side, we demand separation of duty between the two medical examinations, i.e. they have to be performed by different medical officers, such that no single medical officer can manipulate the hiring process by rejecting unwanted candidates for fabricated medical reasons. Regarding confidentiality requirements, we assume that the two medical reports are highly confidential due to their potentially sensitive contents. In particular, they are confidential for the employees in the HR department, who should only be able to access information on a need-to-know basis, e.g. the CVs of applicants.

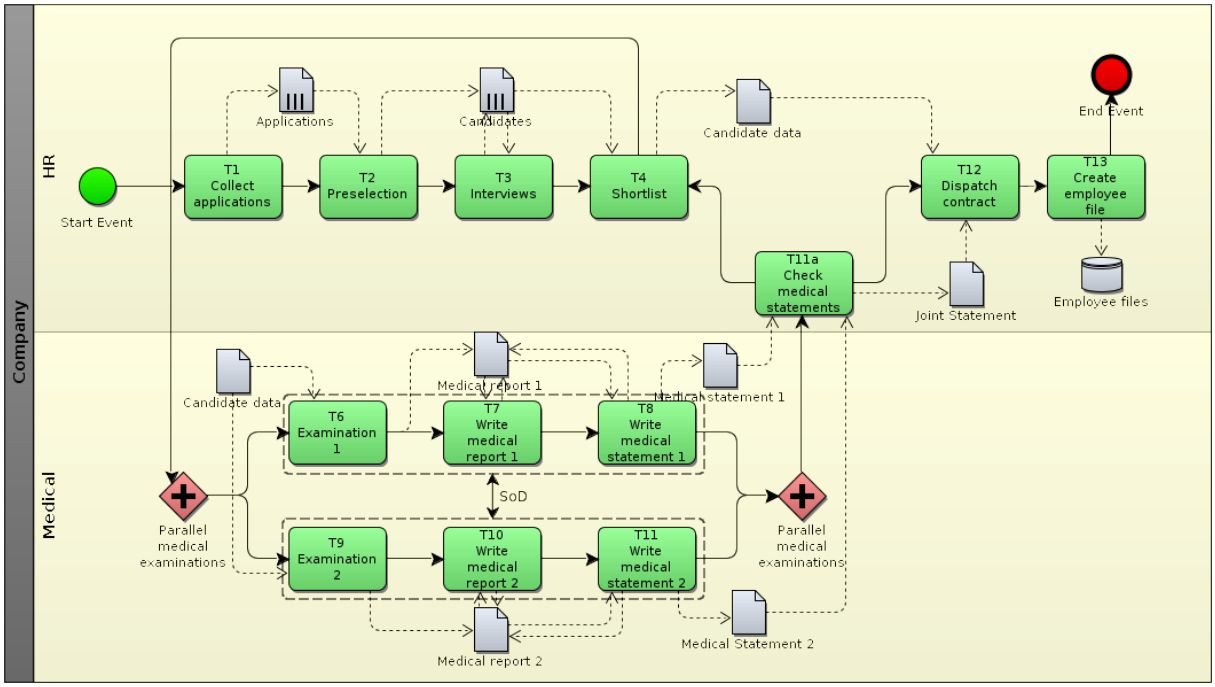

Figure 1: Example workflow, adapted from Brucker and Hang

In general, we assume there is a set of security domains, which are used to classify documents exchanged between activities, and a flow policy that specifies the allowed information flows between domains. We assign a domain to every document (a security classification) and to every employee (a security clearance). These classifications and clearances determine which users are allowed to participate in which activities of the workflow. For example, employees with an *HR* clearance are not allowed to participate in the activities creating the medical report; otherwise, they would have direct access to confidential medical data. We formally define the constraints regarding classifications, clearances and flow policies in Section 3.1.

Besides direct information flows via transfer of documents, we aim to control indirect flows, where confidential information is deducible from observations of the system behaviour. For example, an HR employee in the above workflow can deduce whether the two medical officers agreed about the fitness of the candidate by observing whether the workflow proceeds to activity 12 after the medical examinations or reverts to activity 4. This is acceptable and actually necessary in our scenario, as long as this is the only bit of information about the medical condition of the candidate that can be deduced by HR personnel. Our goal is to verify that the workflow indeed does not leak any additional medical information to non-medical personnel. In the next section, we begin by formally modelling the workflow, and then proceed to formalise the security requirements.

1.2 Preliminaries

We briefly recall the definitions of (state-) event systems and security predicates from the MAKS framework for possibilistic information flow [15] that we use in this paper. An event system $ES = (E, I, O, Tr)$ is essentially a (prefix-closed) set of traces $Tr \subseteq E^*$ that are finite sequences of events in the event set E.

The disjoint sets $I \subseteq E$ and $O \subseteq E$ designate input and output events, respectively. We denote the empty trace as $\langle\rangle$, the concatenation of traces α and β as $\alpha.\beta$, and the projection of a trace α onto a set E as $\alpha|_E$. In the composition $ES_1 \| ES_2$ of two event systems ES_1 and ES_2, input events of one system matching output events of the other system are connected (and vice versa) and thus become internal events of the composed system. The set of traces is the set of interleaved traces of the two systems, synchronised on events in $E_1 \cap E_2$:

$$Tr(ES_1 \| ES_2) = \{\alpha \in (E_1 \cup E_2)^* \mid \alpha|_{E_1} \in Tr(ES_1) \wedge \alpha|_{E_2} \in Tr(ES_2)\}$$

A state-event system $SES = (E, I, O, S, s_0, T)$ has a set of states S, a starting state $s_0 \in S$, and a transition relation $T \subseteq (S \times E \times S)$. The event system *induced* by a state-event system has the same sets of events and the set of traces that is enabled from the starting state via the transition relation.

The MAKS framework defines a collection of basic security predicates (BSPs). Many existing information flow properties from the literature can be expressed as a combination of these BSPs. Each BSP is a predicate on a set of traces with respect to a view $\mathcal{V}$. A view $\mathcal{V} = (V, N, C)$ on an event system $ES = (E, I, O, Tr)$ is defined as a triple of event sets that form a disjoint partition of E. The set V defines the set of events that are visible for an observer, C are the confidential events, and the events in N are neither visible nor confidential. Notable examples for BSPs, that we will use in this paper, are backwards-strict deletion (*BSD*) and backwards-strict insertion of admissible confidential events (*BSIA*)[1], defined in [19] as follows:

$$\begin{aligned} BSD_{\mathcal{V}}(Tr) \equiv & \forall \alpha, \beta \in E^*. \forall c \in C. (\beta.c.\alpha \in Tr \wedge \alpha|_C = \langle\rangle) \\ & \Rightarrow \exists \alpha' \in E^*. \left(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.\alpha' \in Tr\right) \\ BSIA_{\mathcal{V}}(Tr) \equiv & \forall \alpha, \beta \in E^*. \forall c \in C. (\beta.\alpha \in Tr \wedge \alpha|_C = \langle\rangle \wedge \beta.c \in Tr) \\ & \Rightarrow \exists \alpha' \in E^*. \left(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.c.\alpha' \in Tr\right) \end{aligned}$$

Intuitively, the former requires that the occurrence of confidential events must not be deducible, while the latter requires that the *non*-occurrence of confidential events must not be deducible. Technically, they are closure properties of sets of traces. For example, if a trace in Tr contains a confidential event, then *BSD* requires that a corresponding trace without the confidential event exists in Tr that yields the same observations. This means the two traces must be equal with respect to visible V-events, while N-events might be adapted to correct the deletion of the confidential event.

2 System model

In order to verify that a workflow satisfies given security requirements, we need a formal model of workflows and their behaviour. We first define our notion of workflows. For simplicity, we omit aspects such as exceptions or compensation handling, but our definition suffices for our purpose of discussing the verification of security requirements for workflows.

Definition 1. *A workflow $W = (\mathcal{A}, Docs, SF, MF, U)$ consists of*

- *a set $\mathcal{A}$ of activities,*

[1] In [19], *BSIA* is defined with an additional parameter ρ that allows to strengthen the property by further specifying positions at which confidential events must be insertable. For simplicity, we choose to fix this parameter to ρ_E in the notation of [19], i.e we only require confidential events to be insertable into a trace without interfering with observations if they are in principle admissible exactly at that point in the trace.

- *a set Docs of data items,*
- *a set* $SF \subseteq (\mathcal{A} \times \mathcal{A})$ *of sequence flows, where* $(a_1, a_2) \in SF$ *represents the fact that upon completion of* a_1*, it may send a trigger to* a_2 *signalling it to start execution, and*
- *a set* $MF \subseteq (\mathcal{A} \times Docs \times \mathcal{A})$ *of message flows, where* $(a_1, d, a_2) \in MF$ *represents data item* d *being an output of activity* a_1 *and an input to* a_2*, and*
- *a set* U *of users participating in the workflow.*

The sets $\mathcal{A}$ and $Docs$ correspond to the nodes of a workflow diagram such as Figure 1, while SF and MF correspond to the solid and dashed edges, respectively.

We define the behaviour of workflows, not in a monolithic way, but in terms of the behaviours of components representing activities communicating with each other. As we will show in Section 4, this simplifies the verification, because it allows us to use the decomposition methodology of [13] to verify the security of the overall system by verifying security properties of the subcomponents. We believe that such a decomposition approach can help in scaling up verification of information flow properties to larger systems.

Each activity a is therefore modelled as a state-event system $SES_a = \left(E_a, I_a, O_a, S_a, s_a^0, T_a\right)$ analogously to Definition 3 of [13]. The set of events E_a consists of events of the form

- $Start_a(u)$, starting the activity a and assigning it to the user $u \in U$,
- $End_a(u)$, marking the end of the activity,
- $Send_a(a', msg)$ and $Recv_a(a', msg)$, representing the sending (or receiving, respectively) of a message msg from activity a to activity a' (or vice versa),
- $Setval_a(u, i, val)$ and $Outval_a(u, i, val)$, representing a user $u \in U$ reading (or setting, respectively) the value val of data item i, and
- a set of internal events τ_a.

We denote the set of events of a given activity $a \in \mathcal{A}$ as E_a, and the set of all events in a workflow as $E_W = \bigcup_{a \in \mathcal{A}} E_a$. We denote the set of events of a given user $u \in U$ as $E_u = \{Start_a(u) \mid a \in \mathcal{A}\} \cup \{Setval_a(u, i, val) \mid a \in \mathcal{A}, i \in Docs, val \in Val\} \cup \{Outval_a(u, i, val) \mid a \in \mathcal{A}, i \in Docs, val \in Val\} \cup \{End_a(u) \mid a \in \mathcal{A}\}$, and the set of all user interaction events as $E_U = \bigcup_{u \in U} E_u$. The messages between activities can have the form

- $Trigger$, used to trigger a sequence flow to a successor activity in the workflow,
- $Data(i, v)$, used to transfer the value v for data item i, and
- $AckData(i)$, used to acknowledge the receipt of a data item.

Using separate messages for data and sequence flows is inspired by the BPMN standard, which describes its (informal) execution semantics in terms of tokens that are passed from one activity to the next, representing control flow separately from data flows. In addition, this separation simplifies the modelling of confidentiality, as it becomes straightforward to classify events transporting $Data$ messages into confidential or non-confidential events based on the classification of the data items they transport.

The local states of the activities include program variables such as a program counter and a mapping $Mem\colon Docs \to Val$, storing the values of data items. After initialisation, the activity waits for messages from other activities, transferring input data or triggering a sequence flow. When one (or more) of the incoming transitions have been triggered, the activity internally computes output messages (possibly via interaction with users), sends them via the outgoing data associations, and triggers outgoing sequence flows. In Appendix A, we formally specify two types of activities as examples, namely user activities that allow users to read and write data items, and gateway activities that make a decision on the control flow based on the contents of their input data items.

Each of these state-event systems SES_a induces a corresponding event system ES_a. The overall system then emerges from the composition of these event systems ES_a for every activity $a \in \mathcal{A}$, together

with a communication platform ES_P:

$$ES_W = (\|_{a \in \mathcal{A}} ES_a) \| ES_P$$

We call ES_W the *workflow system* for the workflow W. We reuse the communication platform of [13], which is formally specified in Section 2.3 of [13]. It asynchronously forwards messages between the activities. As we do not assume that it provides guarantees regarding message delivery, its specification is very simple.[2] Upon composition with the platform, the communication events between the activities become internal events of the composed system. Only the communication events with users remain input and output events. These events form the user interface of the workflow system.

A simple version of our example workflow can be represented as a composition of instances of the activity types specified in Appendix A. We can represent the activity T11a in Figure 1 as a gateway that decides on the control flow based on the results of the medical examinations: If they are positive, the workflow continues with dispatching the contract, otherwise it goes back to selecting another candidate from the shortlist. The other activities essentially consist of users reading and generating documents, so we can represent them as user activities. Of course, these activities can be enriched with further details, e.g. the interviews can be expanded to subprocesses of their own, but we assume that this is handled in a subsequent refinement step and consider only the abstract level in this paper.

3 Security policies

3.1 Confidentiality

We assign security domains from a set $\mathcal{D}$ of domains to the data items exchanged between the activities of the workflow. We denote this domain assignment function by $dom\colon Docs \to \mathcal{D}$. A flow policy is a reflexive and transitive relation on domains and specifies from which domains to which other domains information may flow.[23] Note that, even though we focus on confidentiality in this paper, also integrity requirements can be seen as a dual to confidentiality and handled using information flow control. For example, in [23] a lattice of combined security levels is built as a product of a confidentiality lattice and an integrity lattice. For our example workflow, we only require two confidentiality domains HR and Med. The medical reports $MedReport1$ and $MedReport2$ created by activities T6–T8 and T9–T11 in Figure 1 are assigned to the Med confidentiality domain, and other data items to the HR domain. The example flow policy states that information may flow from HR to Med, but not vice versa, i.e. $HR \rightsquigarrow Med$ and $Med \not\rightsquigarrow HR$ (see Figure 2).

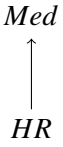

Figure 2: Flow policy

Users read and write the contents of data items via the inputs and outputs of activities they participate in. In order to exclude unwanted direct information flows, we have to make sure that the classifications of the data items that users work with are compatible with their clearances. A straightforward approach is to enforce a Bell-LaPadula style mandatory access control. This can be formulated in terms of classifications that are assigned to activities based on the classifications of their inputs and outputs:

Definition 2. *An activity classification $cl_{\mathcal{A}}\colon \mathcal{A} \to \mathcal{D}$ is an assignment of domains to activities such that*

[2] However, it is possible to adapt the decomposition methodology to other communication models, e.g. providing some notion of reliability of message delivery or means for synchronous communication. For example, see Appendix A for some remarks on guaranteeing a notion of ordered message delivery.

1. *for all input data items i of an activity a,* $dom(i) \rightsquigarrow cl_{\mathcal{A}}(a)$*, and*
2. *for all output data items i of an activity a that may be assigned to untrusted users,* $cl_{\mathcal{A}}(a) \rightsquigarrow dom(i)$.

We allow users to participate in an activity of a given classification only if they have a matching clearance. We denote the mapping of users to clearances as $cl_U : U \to \mathcal{D}$. The conditions in Definition 2 correspond to the Simple Security and $*$-Property of the Bell-LaPadula model, respectively. Note that we relax the $*$-Property by allowing trusted users to downgrade data items. Otherwise, we would not be able to assign a classification to the activities T8 and T11 in our example workflow, because they have high inputs (the medical reports) and low outputs (the statements about the final result of examinations). However, this specific flow of information in the example is acceptable and necessary, because the output should contain only the non-confidential final decision of the medical officer[3] required by the HR department, while the detailed content of the medical reports remains classified as confidential. Essentially, we admit inputs and outputs of trusted users to act as a channel for declassification that is not formally controlled by our information flow analysis. It would be possible to model declassification more explicitly, e.g. using intransitive flow policies [17], but for simplicity we choose this solution for this paper. The same approach is followed in [33], for example. See [32] for an early discussion of this approach to downgrading and [28] for a general overview of principles and dimensions of declassification.

Regardless of whether trusted users are present or not, we want to verify that the system itself does not leak information about data items i with classification $dom(i) \not\rightsquigarrow d$ to users with clearance d. The set of confidential events for a domain d thus consists of events setting or reading values of these data items, while events of activities whose classification is allowed to flow into d are considered to be potentially observable for users in domain d:

Definition 3. *Let* $d \in \mathcal{D}$ *be a domain. The security view on a workflow system* ES_W *for* d *is defined as* $\mathcal{V}_d = (V_d, N_d, C_d)$*, where*

$$
\begin{aligned}
V_d &= \bigcup_{cl_{\mathcal{A}}(a) \rightsquigarrow d} E_a \\
C_d &= \{Setval_a(u,i,val) \mid \exists u \in U, i \in Docs, v \in Val.\ dom(i) \not\rightsquigarrow d \wedge cl_{\mathcal{A}}(a) \not\rightsquigarrow d\} \\
&\quad \cup \{Outval_a(u,i,val) \mid \exists u \in U, i \in Docs, v \in Val.\ dom(i) \not\rightsquigarrow d \wedge cl_{\mathcal{A}}(a) \not\rightsquigarrow d\} \\
N_d &= E \setminus (V_d \cup C_d)
\end{aligned}
$$

The set C_d contains the confidential input and output events.[4] Note that we assume that confidential information enters the system only via user input or output, and that the system does not generate confidential information by itself (e.g. by generating cryptographic key material). If that were the case, the corresponding system events would have to be added to the set C_d. Moreover, it is worth pointing out that we consider certain other types of information to be *non-confidential*. In particular, the information whether an activity has been performed or not, or the information which user has performed which activity is considered to be non-confidential. Again, such requirements could be captured by formulating the security view accordingly. For our setting, the above view reflects our security requirement that the values of confidential data items should be kept secret. Hence, we use this view for the rest of this paper.

In Section 4, we describe how to verify that a given workflow system satisfies the security predicate $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$ with respect to this view for every domain d. It expresses that confidential user inputs and outputs can be deleted or inserted without interfering with the observations of users in domain d.

[3] Which can be further enforced by allowing only Boolean values as content of the low output.

[4] The set C_d only contains events of activities a with $cl_{\mathcal{A}}(a) \not\rightsquigarrow d$, because activities with $cl_{\mathcal{A}}(a) \rightsquigarrow d$ are considered to be visible, and the set of visible and confidential events must be disjoint.

3.2 Separation of duties

As discussed in the introduction, separation of duties is another common security requirement in workflow management systems. Separation of duties can be formally defined as a safety property [3]. The "bad thing" happens when the same user participates in two activities constrained by separation of duty, hence we only allow traces where this does not occur.

Definition 4. *Let $a, a' \in \mathcal{A}$ be two activities. We call the set of traces*

$$\left\{\alpha \in E_W^* \mid \forall u, u' \in U.\ \forall e_1, e_2 \in \alpha.(e_1 \in (E_a \cap E_u) \wedge e_2 \in (E_{a'} \cap E_{u'})) \rightarrow u \neq u'\right\}$$

a separation-of-duty property $P_{SoD}^{a,a'}$.

As we have modelled user assignment explicitly as events, this property can also be characterised by requiring that 1. constrained activities are assigned to different users, and 2. users may participate in an activity only after they have been assigned to it:

$$\begin{aligned} P_{SoD}^{a,a'} \supseteq &\left\{\alpha \in E_W^* \mid \forall u, u' \in U.\ Start_{a_1}(u) \in \alpha \wedge Start_{a_2}(u') \in \alpha \longrightarrow u \neq u'\right\} \\ &\cap \left\{\alpha \in E_W^* \mid \forall a \in \mathcal{A}, u \in U, e \in (E_a \cap E_u).\ Start_a(u) \notin \alpha \longrightarrow e \notin \alpha\right\} \end{aligned}$$

A system with a set of traces Tr and events E satisfies such a property iff $Tr \subseteq P_{SoD}^{a,a'}$. In our example workflow, there are separation of duty constraints between the activities belonging to the two medical examinations ($T6$–8 in Figure 1 one the one hand, and $T9$–11 on the other hand). Hence, we want to enforce $P_{SoD}^{a,a'}$ for the pairs $(a, a') \in \{T6, T7, T8\} \times \{T9, T10, T11\}$.

Similarly, other runtime-enforceable security policies [30] can be modelled as safety properties. In this paper, we focus on the above notion of separation of duty as an example and investigate its relation to information flow in Section 4.2.

4 Verification

4.1 Information flow security

To ease the verification of the security of a workflow system, we decompose it into the individual activities of the workflow and make use of the methodology presented in [13] to verify the resulting distributed system. For each domain $d \in \mathcal{D}$, we verify that users in that domain can learn nothing about information that is confidential for them. The first step of the methodology [13] is to partition the activities into a set of low activities $\mathcal{A}_L^d = \{a \in \mathcal{A} \mid cl_{\mathcal{A}}(a) \rightsquigarrow d\}$ that are (potentially) visible in domain d and a set of high activities $\mathcal{A}_H^d = \{a \in \mathcal{A} \mid cl_{\mathcal{A}}(a) \not\rightsquigarrow d\}$ that are not visible and may handle confidential information.[5] It follows that $\mathcal{V}_d$ from Definition 3 is a global security view as defined in [13, Definition 13], i.e. the visible events are exactly the events of the low activities, the set of confidential events is a subset of the events of the high activities, and the remaining events are non-visible and non-confidential.

The second step is finding suitable local views $\mathcal{V}_d^a$ for high activities $a \in \mathcal{A}_H^d$ in order to verify that they do not leak confidential information to low activities. Hence, we cannot generally treat communication events of these activities as N-events, as we did in the global view, but we have to consider some of them as V-events (e.g. a high activity sending a trigger or a declassified data item to a low activity)

[5] In [13], the set $\mathcal{A}_L^d$ is called the set of observers, while $\mathcal{A}_H^d$ is called the set of friends. This might be a bit counterintuitive in our setting for some readers, as the friends would be the activities that are *not* visible. To avoid confusion, we simply speak of low and high activities, respectively.

and some of them as C-events (e.g. a high activity receiving a confidential data item). Intuitively, this means we split each of these activities into a part that visibly interacts with low activities and a part that handles confidential data, and verify that the latter does not interfere with the former. Technically, these local views satisfy certain constraints that allow us to instantiate the compositionality result of [13], as we discuss below.

Definition 5. *Let $d \in \mathcal{D}$ be a domain, and $a \in \mathcal{A}_H^d$ be a high activity for d. Furthermore, let $Docs_d^C = \{i \in Docs \mid dom(i) \not\rightsquigarrow d\}$ denote the set of data items that are confidential for d. The* local view for *a is defined as $\mathcal{V}_d^a = (V_d^a, N_d^a, C_d^a)$ with*

$$V_d^a = (I_a \cup O_a) \setminus \bigcup_{i \in Docs_d^C} E_i$$

$$C_d^a = \bigcup_{i \in Docs_d^C} (E_i \setminus \{Send_a(b,m) \mid \exists v.\, m = Data(i,v) \vee m = AckData(i)\})$$

$$N_d^a = E_a \setminus (V_d^a \cup C_d^a)$$

where the set E_i of high communication events containing data item i is defined as

$$E_i = \big\{ e \mid \exists b \in \mathcal{A}_H^d, m, u, v.\ (m = Data(i,v) \vee m = AckData(i)) \\ \wedge (e = Send_a(b,m) \vee e = Recv_a(b,m) \vee e = Setval_a(u,i,v) \vee e = Outval_a(u,i,v)) \big\}$$

Combining these local views, we define the composed view for *d as $\mathcal{V}_{d^+} = (V_{d^+}, N_{d^+}, C_{d^+})$ where*

$$V_{d^+} = \bigcup_{a \in \mathcal{A}_H^d} V_d^a \cup \bigcup_{a \in \mathcal{A}_L^d} E_a \qquad C_{d^+} = \bigcup_{a \in \mathcal{A}_H^d} C_d^a \qquad N_{d^+} = E_W \setminus (V_{d^+} \cup C_{d^+})$$

Note that the combined view $\mathcal{V}_{d^+}$ is *stronger* than our global view $\mathcal{V}_d$ in the sense that more events are considered confidential or visible for an observer in domain d. Theorem 1 of [19] tells us that $BSD_{\mathcal{V}_{d^+}} \wedge BSIA_{\mathcal{V}_{d^+}}$ for the stronger view implies $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$.

Also note that all communication events with low activities are considered visible, and that the forwarding of confidential data items from one high activity to another is considered non-confidential. The justification for this is that secrets enter and leave the subsystem of high activities through communication with users and low activities, and the forwarding between high activities can be considered as internal processing. Hence, we can use communication events between high activities for correcting perturbations caused by inserting or removing confidential user inputs. We make use of this fact in the proof of the following theorem, which states the security of activities as we have specified them in Appendix A in terms of the transition relations T_a^{gen}, T_a^{user} and $T_a^{gw(Cond)}$.

Theorem 1. *Let W be a workflow, $d \in \mathcal{D}$ a domain and SES_a for $a \in \mathcal{A}$ an activity. If the transition relation of SES_a is*

- *$T_a^{gen} \cup T_a^{user}$, or*
- *$T_a^{gen} \cup T_a^{gw(Cond)}$ and Cond does not depend on confidential data for d,*

then $BSD_{\mathcal{V}_d^a}(Tr_a) \wedge BSIA_{\mathcal{V}_d^a}(Tr_a)$ holds.

The proof of this and the following theorems can be found in the extended version of this paper [5]. We use the unwinding technique [16] for the proof. Note that since the generic transition relation T_a^{gen} and the activity-specific transition relations are disjoint, we can partition this proof into a generic part that covers the events and states used in T_a^{gen}, and an activity-specific part. Therefore, if we want to use a different kind of activity than the ones specified in this paper, and we reuse the generic part T_a^{gen} of the transition relation, then we can also reuse most of this proof.

The next step is to instantiate the compositionality result of [13], which states that the security of the overall system with respect to the global security view is implied by the security of the subsystems with respect to their local views. However, our local views do not quite satisfy the requirement of being *C-preserving* in the sense of Definition 18 of [13], because that definition disallows *N*-events in the communication interface between subsystems. Hence, we slightly adapt the notion C-preserving views, allowing *Send* events to be in *N*:

Definition 6. *Let $\mathcal{A}_H^d \subseteq \mathcal{A}$ and $C \subseteq E_{\mathcal{A}_H^d}$. A family $(\mathcal{V}^a)_{a\in\mathcal{A}_H^d}$ of views $\mathcal{V}^a = (V_a, N_a, C_a)$ for E_a is* C-preserving for *C iff*

1. *$a \in \mathcal{A}_H^d$ and $b \notin \mathcal{A}_H^d$ implies $\forall m.Send_a(b,m) \in V_a \wedge Recv_a(b,m) \in V_a$.*
2. *$a, a' \in \mathcal{A}_H^d$ implies*
   (a) *$Recv_{a'}(a,m) \in C_{a'}$ iff $Send_a(a',m) \notin V_a$ and*
   (b) *$Recv_{a'}(a,m) \in V_{a'}$ iff $Send_a(a',m) \in V_a$*
3. *$C \cap E_a \subseteq C_a$ for all $a \in \Phi$.*

As can be easily seen, our local views are C-preserving for the set of global confidential events C_d from Definition 3: communication with low activities is visible, corresponding *Recv* and *Send* events are either visible or non-visible (where non-visible *Recv* events need to be confidential, while the corresponding *Send* events are allowed to be treated as *N*-events), and events that are confidential in the global view are confidential for the local views.

It turns out that the compositionality result of [13] still holds for our weakened notion of C-preserving local views; a sufficient (but not necessary) condition is that the subsystems satisfy not only *BSD* (as in [13]), but *BSD* and *BSIA*, which our activities happen to do.

Theorem 2. *Let W be a workflow, $ES_W = (\|_{a\in\mathcal{A}} ES_a) \| ES_P$ be a workflow system, $\mathcal{V}_d$ be a global security view for domain d, and $\left(\mathcal{V}_d^a\right)_{a\in\mathcal{A}_H^d}$ be a family of local views that is C-preserving for C_d. If for all $a \in \mathcal{A}_H^d$, ES_a satisfies $BSD_{\mathcal{V}_d^a} \wedge BSIA_{\mathcal{V}_d^a}$, then ES_W satisfies $BSD_{\mathcal{V}_{d^+}} \wedge BSIA_{\mathcal{V}_{d^+}}$ and, therefore, $BSD_{\mathcal{V}_d} \wedge BSIA_{\mathcal{V}_d}$.*

Note that, if other kinds of activities than the ones from Appendix A should be part of the workflow, it is only required to prove that their specifications also satisfy the security predicates for the local views, in order to show that the overall workflow satisfies the information flow security predicates.

We have formalised and verified our model and proofs using the interactive theorem prover Isabelle [24]. Our development is based on a formalisation of the MAKS framework developed by the group of Heiko Mantel at TU Darmstadt (unpublished as of this writing). We intend to make our formalisation publicly available when the MAKS formalisation is released.

Conceptually, the main difference between our workflow management systems and the shopping mall system described in [13] lies in the relation between users and the system. In the shopping scenario, there is a one-to-one correspondence between users and software agents running in the system. Communication with the users happens only during initialisation, when users write their preferences into the initial memory of their agents, which run autonomously thereafter. In our workflow systems, the interaction is much more dynamic, as multiple activities can be assigned to the same user at runtime and there is ongoing communication between users and the system. This has impact on the system model — we introduced additional events for user interaction — and the construction of views. The partitioning into high and low activities is based on classifications of data items and activities, and access control has to ensure that only users with a matching clearance can participate in an activity, so that our security views are actually in line with the possible runtime observations of users. Despite these differences, we have seen that the methodology of [13] can be applied with small technical adjustments.

4.2 Compatibility with separation of duties

As described in Section 3.2, we can formalise constraints such as separation of duty as safety properties. Having established information flow security of our workflow system, we now ask whether these security properties are preserved when enforcing separation of duty constraints. In general, this is not the case. Altering a system such that it satisfies a safety property can be seen as a refinement, and it is well-known that possibilistic information flow security is not preserved under refinement in general [18]. Consider, for example, the security predicate *BSIA*. Repeatedly inserting confidential events of different users into a trace can exhaust the possible user assignments that would satisfy the separation of duty constraints, thus deadlocking the process and making further visible observations impossible. We can, however, try to find sufficient conditions under which information flow properties are preserved:

Theorem 3. *Let $ES = (E, I, O, Tr)$ be an event system and $\mathcal{V} = (V, N, C)$ be a view for ES. Let $E_a, E'_a \subseteq E$ be two disjoint sets of events corresponding to activities a and a', and let $P_{SoD}^{a,a'}$ be an SoD property. Let $E_u \subseteq V \cup C$ be the communication events with a user u and $E_U = \bigcup_{u \in U} E_u$ the set of all user events. If*

1. *user assignment is non-confidential, i.e. there is a set $E^{assign} \subseteq E \setminus C$ of assignment events, and a user u may only participate in an activity after having been assigned to it via an event from $E^{assign} \cap E_u$, or*
2. *only confidential or only visible user I/O events of activities a and a' are enabled in ES, i.e. there is a set $E^{disabled} \subseteq E$ of events that never occur in a trace of ES, and $V \cap (E_a \cup E'_a) \cap E_U \subseteq E^{disabled}$ or $C \cap (E_a \cup E'_a) \cap E_U \subseteq E^{disabled}$ holds, or*
3. *the SoD constraint between a and a' is already enforced by ES, i.e. $Tr \subseteq P_{SoD}^{a,a'}$,*

then $BSD_{\mathcal{V}}(Tr) \wedge BSIA_{\mathcal{V}}(Tr)$ implies $BSD_{\mathcal{V}}(Tr \cap P_{SoD}^{a,a'}) \wedge BSIA_{\mathcal{V}}(Tr \cap P_{SoD}^{a,a'})$.

In our running example, we can choose $E^{assign} = \{Start_a(u) \mid u \in U\}$ and apply the first case of the theorem for the workflow system ES_W and a view $\mathcal{V}_{d^+}$, because only the details of the results of the medical examinations are confidential, not the information who carried out the examinations. Furthermore, in case $cl_{\mathcal{A}}(a) \neq cl_{\mathcal{A}}(a')$, the mandatory access control described in Section 3.1 already enforces SoD statically, so the third condition also applies. In general, Theorem 3 gives us sufficient conditions for the compatibility of SoD constraints and information flow properties, taking into account the classifications of events that are relevant for enforcing SoD. Similar results could be developed for other classes safety properties that are of interest in workflows, but we leave this as future work. Note that Theorem 3 is not specific to workflow systems as specified in this paper. It can be applied to any system where users perform different activities in the presence of separation of duty constraints.

5 Related work

We build upon the MAKS framework for possibilistic information flow control [15], which is suitable for formulating and verifying information flow policies at the specification level. We have focused on confidentiality of data from unauthorised employees within the organisation, but in principle information flow control can be adapted to different attacker models and security policies by choosing the security views appropriately. Furthermore, approaches have been proposed to take into account factors such as communication over the Internet [12] or encrypted communication channels [14]. In [25], a connection between role-based access control (RBAC) and mandatory access control is drawn, which might be adapted to enforce the mandatory access control we described in Section 3.1 using RBAC mechanisms.

Early examples for workflow management systems with distributed architectures include [2, 22, 31]. Later, computing paradigms with a similar spirit have emerged, e.g. service-oriented architectures or cloud computing. We see these techniques and standards as complementary to our work, as they can be used for the implementation of our abstract specifications.

BPMN extensions to annotate business process diagrams with security annotations can be found in [6, 26, 33]. Closest to the security requirements considered by us comes the notation proposed in [33] that supports both the annotation of activities with separation of duty constraints and the annotation of documents and process lanes with confidentiality and integrity classifications or clearances, respectively.

Several proposals for a formal semantics of workflow specifications can be found in the literature. For example, [34] maps BPMN diagrams to CSP processes and describes how the formal semantics can be leveraged to compare and analyse workflow diagrams, e.g. with respect to consistency. It focuses on the control flow and does not model data flows. In [35], workflows are represented as statements in a workflow description language, which is mapped to a representation as hierarchical state machines. An information flow analysis algorithm is described, but the actual information flow property that it checks is not stated in a declarative, mechanism-independent way. [1] represents workflows as Petri nets and describes an approach for information flow analysis. The focus is on keeping the occurrence of tasks confidential, whereas our work focuses on the confidentiality of the data that is processed in the workflow. In [4] and [29], workflows are formalised as transition systems and model-checking is employed to verify properties specified as LTL formulas. This is suitable to verify safety or liveness properties, whereas the information flow predicates considered by us can be seen as hyperproperties [8].

6 Conclusion

Graphical notations such as BPMN are widely used for workflow specification. We have presented an approach to formally model both the behaviour of a workflow and the associated security requirements, and described how to apply the decomposition methodology of [13] and how to verify a distributed workflow management system with ongoing user interaction. We have shown that, even though possibilistic information is in general not refinement-closed, the enforcement of separation of duty is compatible with the information flow security of the system under certain assumptions.

We have sketched how a simple version of our example workflow can be represented as a composition of instantiations of the activity types specified in Appendix A. As we have shown the security of these activities in Theorem 1, we can use Theorem 2 to derive the security of the composed system from the security properties of the individual activities. This demonstrates how instantiations of a type of activities that has been proven secure once can be plugged into larger workflows in a secure way. Hence, we believe that this compositional approach can help in making verification techniques for information flow scale to larger workflow systems. However, more work is needed before this approach can actually be applied to realistic systems. For example, tool support for translating a more realistic subset of BPMN to our system model would be a major step in this direction, which would also help us evaluate our approach with a sample of existing workflows.

Moving from an abstract specification towards the implementation level is another important direction of future work. This paper deals with workflows on a high level of abstraction. We intend to work on notions of security-preserving refinement that allow us to expand abstract activities in a workflow into more concrete subprocesses and refine the behaviour of atomic activities towards an executable implementation. There is a large body of existing work that we can build upon for this purpose, such as action refinement for replacing atomic events on the abstract level with sequences of more concrete events [11],

switching between event-based and language-based notions of information flow [20], or directly generating executable code from specifications [10]. In the long term, we hope that these decomposition and refinement techniques will contribute to making the step-wise development of secure workflow systems from workflow diagrams to executable code more scalable and efficient.

Acknowledgements We thank Richard Gay, Sylvia Grewe, Steffen Lortz, Heiko Mantel and Henning Sudbrock for providing a formalisation of the MAKS framework in Isabelle/HOL that allowed us to verify our main results in Isabelle, and the anonymous reviewers for helpful comments on the paper.

References

[1] Rafael Accorsi & Andreas Lehmann (2012): *Automatic Information Flow Analysis of Business Process Models*. In: *BPM*, pp. 172–187, doi:10.1007/978-3-642-32885-5_13.

[2] Gustavo Alonso, Roger Günthör, Mohan Kamath, Divyakant Agrawal, Amr El Abbadi & C. Mohan (1996): *Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients*. *Distributed and Parallel Databases* 4(3), pp. 229–247, doi:10.1007/BF00140951.

[3] Bowen Alpern & Fred B. Schneider (1987): *Recognizing safety and liveness*. *Distributed Computing* 2(3), pp. 117–126, doi:10.1007/BF01782772.

[4] Wihem Arsac, Luca Compagna, Giancarlo Pellegrino & Serena Elisa Ponta (2011): *Security Validation of Business Processes via Model-Checking*. In: *Engineering Secure Software and Systems*, *LNCS* 6542, Springer, pp. 29–42, doi:10.1007/978-3-642-19125-1_3.

[5] Thomas Bauereiss & Dieter Hutter (2013): *Possibilistic information flow security of workflow management systems*. Technical Report. Available at `http://bauereiss.name/papers/WorkflowSecurity_TR.pdf`.

[6] Achim D. Brucker, Isabelle Hang, Gero Lückemeyer & Raj Ruparel (2012): *SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes*. In: *SACMAT 2012*, ACM, pp. 123–126, doi:10.1145/2295136.2295160.

[7] David D. Clark & David R. Wilson (1987): *A Comparison of Commercial and Military Computer Security Policies*. *IEEE Symposium on Security and Privacy*, pp. 184–194, doi:10.1109/SP.1987.10001.

[8] Michael R. Clarkson & Fred B. Schneider (2010): *Hyperproperties*. *Journal of Computer Security* 18(6), pp. 1157–1210, doi:10.3233/JCS-2009-0393.

[9] Riccardo Focardi & Roberto Gorrieri (1995): *A Classification of Security Properties for Process Algebras*. *Journal of Computer Security* 3(1), pp. 5–33, doi:10.3233/JCS-1994/1995-3103.

[10] Florian Haftmann & Tobias Nipkow (2007): *A code generator framework for Isabelle/HOL*. In: *Theorem Proving in Higher Order Logics: Emerging Trends*. Available at `http://es.cs.uni-kl.de/events/TPHOLs-2007/proceedings/B-128.pdf`.

[11] Dieter Hutter (2006): *Possibilistic Information Flow Control in MAKS and Action Refinement*. In: *ETRICS*, *LNCS* 3995, Springer, pp. 268–281, doi:10.1007/11766155_19.

[12] Dieter Hutter (2007): *Preserving Privacy in the Web by Using Information Flow Control*. In Andreas U. Schmidt, Michael Kreutzer & Rafael Accorsi, editors: *Long-Term and Dynamical Aspects of Information Security: Emerging Trends in Information and Communication Security*, Nova Science.

[13] Dieter Hutter, Heiko Mantel, Ina Schaefer & Axel Schairer (2007): *Security of multi-agent systems: A case study on comparison shopping*. *Journal of Applied Logic* 5(2), pp. 303–332, doi:10.1016/j.jal.2005.12.015.

[14] Dieter Hutter & Axel Schairer (2004): *Possibilistic Information Flow Control in the Presence of Encrypted Communication*. In: *ESORICS*, *LNCS* 3193, Springer, pp. 209–224, doi:10.1007/978-3-540-30108-0_13.

[15] Heiko Mantel (2000): *Possibilistic Definitions of Security - An Assembly Kit*. In: *CSFW*, IEEE Computer Society, pp. 185–199, doi:10.1109/CSFW.2000.856936.

[16] Heiko Mantel (2000): *Unwinding Possibilistic Security Properties*. In: *ESORICS*, *LNCS* 1895, Springer, pp. 238–254, doi:10.1007/10722599_15.

[17] Heiko Mantel (2001): *Information Flow Control and Applications - Bridging a Gap*. In: *FME*, *LNCS* 2021, Springer, pp. 153–172, doi:10.1007/3-540-45251-6_9.

[18] Heiko Mantel (2001): *Preserving Information Flow Properties under Refinement*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 78–91, doi:10.1109/SECPRI.2001.924289.

[19] Heiko Mantel (2002): *On the Composition of Secure Systems*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 88–101, doi:10.1109/SECPRI.2002.1004364.

[20] Heiko Mantel & Andrei Sabelfeld (2003): *A Unifying Approach to the Security of Distributed and Multi-Threaded Programs*. *Journal of Computer Security* 11(4), pp. 615–676. Available at `http://iospress.metapress.com/content/r0pr0ma4kv8wa542/`.

[21] J. McLean (1996): *A general theory of composition for a class of "possibilistic" properties*. *IEEE Transactions on Software Engineering* 22(1), pp. 53–67, doi:10.1109/32.481534.

[22] Peter Muth, Dirk Wodtke, Jeanine Weissenfels, Angelika Kotz Dittrich & Gerhard Weikum (1998): *From Centralized Workflow Specification to Distributed Workflow Execution*. *Journal of Intelligent Information Systems* 10(2), pp. 159–184, doi:10.1023/A:1008608810770.

[23] Andrew C. Myers, Andrei Sabelfeld & Steve Zdancewic (2006): *Enforcing Robust Declassification and Qualified Robustness*. *Journal of Computer Security* 14(2), pp. 157–196. Available at `http://iospress.metapress.com/content/EYT2D3ERKY3A2H25`.

[24] Tobias Nipkow, Lawrence C Paulson & Markus Wenzel (2002): *Isabelle/HOL: a proof assistant for higher-order logic*. *LNCS* 2283, Springer.

[25] Sylvia Osborn, Ravi Sandhu & Qamar Munawer (2000): *Configuring role-based access control to enforce mandatory and discretionary access control policies*. *ACM Trans. Inf. Syst. Secur.* 3(2), p. 85–106, doi:10.1145/354876.354878.

[26] Alfonso Rodríguez, Eduardo Fernández-Medina & Mario Piattini (2007): *A BPMN Extension for the Modeling of Security Requirements in Business Processes*. *IEICE Transactions* 90-D(4), pp. 745–752, doi:10.1093/ietisy/e90-d.4.745.

[27] A. Sabelfeld & A.C. Myers (2003): *Language-based information-flow security*. *IEEE Journal on Selected Areas in Communications* 21(1), pp. 5–19, doi:10.1109/JSAC.2002.806121.

[28] Andrei Sabelfeld & David Sands (2009): *Declassification: Dimensions and principles*. *Journal of Computer Security* 17(5), pp. 517–548, doi:10.3233/JCS-2009-0352.

[29] Andreas Schaad, Volkmar Lotz & Karsten Sohr (2006): *A model-checking approach to analysing organisational controls in a loan origination process*. In David F. Ferraiolo & Indrakshi Ray, editors: *SACMAT*, ACM, pp. 139–149, doi:10.1145/1133058.1133079.

[30] Fred B. Schneider (2000): *Enforceable security policies*. *ACM Trans. Inf. Syst. Secur.* 3(1), p. 30–50, doi:10.1145/353323.353382.

[31] Hans Schuster, Stefan Jablonski, Thomas Kirsche & Christoph Bussler (1994): *A Client/Server Architecture for Distributed Workflow Management Systems*. In: *PDIS*, IEEE Computer Society, pp. 253–256, doi:10.1109/PDIS.1994.331708.

[32] Daniel F. Stork (1975): *Downgrading in a Secure Multilevel Computer System: The Formulary Concept*. Technical Report, DTIC Document. Available at `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA011696`.

[33] Christian Wolter & Christoph Meinel (2010): *An approach to capture authorisation requirements in business processes*. *Requir. Eng.* 15(4), pp. 359–373, doi:10.1007/s00766-010-0103-y.

[34] Peter Y. H. Wong & Jeremy Gibbons (2008): *A Process Semantics for BPMN*. In: *ICFEM*, *LNCS* 5256, Springer, pp. 355–374, doi:10.1007/978-3-540-88194-0_22.

[35] Ping Yang, Shiyong Lu, Mikhail I. Gofman & Zijiang Yang (2010): *Information flow analysis of scientific workflows*. Journal of Computer and System Sciences 76(6), pp. 390–402, doi:10.1016/j.jcss.2009.11.002.

[36] Aris Zakinthinos & E. Stewart Lee (1997): *A General Theory of Security Properties*. In: IEEE Symposium on Security and Privacy, IEEE Computer Society, pp. 94–102, doi:10.1109/SECPRI.1997.601322.

A Specification of activities

In this appendix, we give a formal specification of the behaviour of our activities using PP-statements. In this formalism, the transition relation of a state-event system is specified by listing pre- and post-conditions on the state for each event (see Section 2.1 of [13] for a formal semantics).

$Recv_a(b, Data(i,v))$; **affects:** Mem, $AQueue$
Pre: $pc = 0$, $(b,i,a) \in MF$, $(b,i) \notin AQueue$
Post: $Mem'(i) = v$, $AQueue' = AQueue \cup \{(b,i)\}$

$Send_a(b, AckData(i))$; **affects:** $AQueue$
Pre: $pc = 0$, $(b,i) \in AQueue$
Post: $AQueue' = AQueue \setminus (b,i)$

$Recv_a(b, Trigger)$; **affects:** $TriggeredBy$
Pre: $pc = 0$
Post: $TriggeredBy' = b$

τ_a^{Active}; **affects:** pc
Pre: $pc = 0$, $TriggeredBy \neq \bot$, $AQueue = \emptyset$
Post: $pc' = 1$

$\tau_a^{SendData}$; **affects:** pc, $MQueue$
Pre: $pc = 2$
Post: $pc' = 3$, $MQueue' = \{(b,i) \mid (a,i,b) \in MF \wedge Mem(i) \neq \bot\}$

$Send_a(b, Data(i,v))$;
affects: $MQueue$, $AQueue$
Pre: $pc = 3$, $(b,i) \in MQueue$
Post: $MQueue' = MQueue \setminus \{(b,i)\}$, $AQueue' = AQueue \cup \{(b,i)\}$

$Recv_a(b, AckData(i))$; **affects:** $AQueue$
Pre: $pc = 3$, $(b,i) \in AQueue$
Post: $AQueue' = AQueue \setminus \{(b,i)\}$

$\tau_a^{AckTimeout}$; **affects:** $AQueue$
Pre: $pc = 3$
Post: $AQueue' = \emptyset$

$\tau_a^{SendTriggers}$; **affects:** pc, $SQueue$
Pre: $pc = 3$, $MQueue = \emptyset$, $AQueue = \emptyset$
Post: $pc' = 4$, $SQueue' = \{b \mid (a,b) \in SF\}$

$Send_a(b, Trigger)$; **affects:** $SQueue$
Pre: $pc = 4$, $b \in SQueue$
Post: $SQueue' = SQueue \setminus \{b\}$

Figure 3: PP-statements of generic transition relation T_a^{gen}

We specify the behaviour of our activities in two parts. The PP-statements in Figure 3 specify the *generic* part of the behaviour of activities, i.e. the communication with other activities in order to exchange data items and trigger sequence flows. For this purpose, it maintains program variables $MQueue$ (which data items still have to be sent), $AQueue$ (which data items still have to be acknowledged), $SQueue$ (which triggers still have to be sent), $TriggeredBy$ (whether and from where a trigger has been received), and $User$ (to which user this activity is assigned). The program counters 0, 3 and 4 correspond to the phases of waiting for inputs and triggers, sending outputs, and sending triggers, respectively.

When the program counter reaches 1, an *activity-specific* transition relation takes over in order to perform the actual activity. In our simple example workflow, we only need two kinds of activities, namely

user input/output and gateways (deciding on the control flow based on a condition *Cond* on input data). The latter continues the workflow with that activity b for which $Cond(b, Mem)$ evaluates to true. These two kinds of activities are specified in Figures 4 and 5, respectively. We denote the transition relations induced by the PP-statements in Figures 3, 4, and 5 as T_a^{gen}, T_a^{user}, and $T_a^{gw(Cond)}$, respectively. The overall transition relation of an activity is the union of T_a^{gen} and an activity-specific transition relation.

$Start_a(u)$; **affects:** $User$
Pre: $pc = 1, User = \bot, cl_U(u) = cl_{\mathcal{A}}(a)$
Post: $User' = u$

$Setval_a(u,i,v)$; **affects:** Mem
Pre: $pc = 1, User = u$
Post: $Mem'(i) = v$

$Outval_a(u,i,v)$; **affects:**
Pre: $pc = 1, User = u, Mem(i) = v$

$End_a(u)$; **affects:** pc
Pre: $pc = 1, User = u$
Post: $pc' = 2$

Figure 4: PP-statements of transition relation T_a^{user} for user activities

$Send_a(b, Trigger)$; **affects:** pc
Pre: $pc = 1, Cond(b, Mem) = \top, (a,b) \in SF$
Post: $pc' = 5$

Figure 5: PP-statement of transition relation $T_a^{gw(Cond)}$ for gateways

After completion of the activity has been signalled by setting the program counter to 2, the generic transition relation takes control again and starts sending output data items to the designated receivers. It makes sure that they have been received by waiting for acknowledgements, and afterwards proceeds by sending triggers to the successor activities in the workflow. An exception to this rule is if a receiver fails to send an acknowledgement; in this case the $\tau_a^{AckTimeout}$ event can be used to signal a timeout and proceed with the workflow. This is important for security, because otherwise a confidential activity could block the progress of the workflow by refusing to acknowledge a data item.

Of course, other modelling decisions are possible to solve this problem. As an alternative, we have also modelled and verified a system specification where the communication platform guarantees causal delivery of messages, i.e. messages from one activity to another are always received in the order that they are sent. This would make acknowledgements unnecessary, because an activity could always be sure that a trigger message is received after all data items, if the messages are sent in this order. However, this shifts complexity from the individual activities to the communication platform and the interface, and it turns out that this makes the proof of compositionality more laborious. Essentially, we had to prove an additional security predicate *FCIA* for the platform and the activities together with several additional side conditions on the local views in order to obtain compositionality. In this paper, we therefore present the above model with explicit acknowledgements for simplicity. However, we intend to further investigate the implications of different guarantees provided by the communication platform in future work.

Actor Network Procedures as Psi-calculi for Security Ceremonies *

Cristian Prisacariu †
Dept. of Informatics, University of Oslo, – P.O. Box 1080 Blindern, N-0316 Oslo, Norway.
cristi@ifi.uio.no

The *actor network procedures* of Pavlovic and Meadows are a recent graphical formalism developed for describing security ceremonies and for reasoning about their security properties. The present work studies the relations of the actor network procedures (ANP) to the recent psi-calculi framework. Psi-calculi is a parametric formalism where calculi like spi- or applied-pi are found as instances. Psi-calculi are operational and largely non-graphical, but have strong foundation based on the theory of nominal sets and process algebras. One purpose of the present work is to give a semantics to ANP through psi-calculi. Another aim was to give a graphical language for a psi-calculus instance for security ceremonies. At the same time, this work provides more insight into the details of the ANPs formalization and the graphical representation.

1 Introduction

Actor Network Procedures (ANP) is a formalism introduced in [33] for modeling security ceremonies [12, 38]. Reasoning about security properties of ceremonies is done using the Procedure Derivation Logic, which comes from a line of research on logics for composition of protocols that started with the Protocol Composition Logic [11, 8]. This formalism that we concentrate on in this paper should not be confused with the work with similar purposes from [35]. Both these approaches [35, 33] aim to be used for describing security ceremonies by drawing inspiration from the actor network theory in sociology, where the book [25] gives a good overview.

Security ceremonies are well motivated in [12, 38] with convincing examples. Technically, security ceremonies are meant to extend security protocols by including the human in the formalization and making explicit the environment (and the attacker). A ceremony may also combine protocols. In consequence, a formalism for security ceremonies is expected to be expressive enough to include existing formalisms for protocols as special cases. Such a formalism should offer the possibility to model human behavior related to the ceremony. Since ceremonies would tend to be large, because of all the assumptions that are included explicitly, we expect compositionality to be a main aspect of the formalism, both for design and for reasoning.

For usability purposes a desired formalism for security ceremonies would offer a graphical language for developing the ceremonies, as well as for reasoning. Yet the graphics should be formally grounded, so to have guarantees for the security results obtained. A good example of such formally grounded graphical languages can be the statecharts [17, 19] or the live sequence charts [18], which were intended for describing concurrent and reactive systems.

*This work was partially supported by the project OffPAD with number E!8324 part of the Eurostars program funded by the EUREKA and European Community.

†**Acknowledgements:** I would like to thank Audun Jøsang for introducing me to security ceremonies and for explaining their practical usefulness and the need for an adequate formalism (i.e., graphical, expressive, and with reasoning capabilities).

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

The aim of the actor network procedures [33] is to be graphical, expressive, compositional, and with formal logical reasoning capabilities. In this paper we look more carefully at this formalism and relate it to the psi-calculi framework, which offers solid semantical analysis and possibility for correlations with existing security formalisms coming from the process algebra approach.

Psi-calculi [3] are a semantics framework where various existing calculi can be found as instances. In particular, the spi- and applied-pi calculi [2, 1] are two instances of interest for security protocols. Psi-calculi are though a non-graphical algebraic formalism; but with strong mathematical background. Results and tools of psi-calculi, e.g., involving theorem provers or true concurrency semantics, could be thus used in the setting of actor network procedures. Nevertheless these could easily be hidden from the security ceremony developer behind the graphical formalism ANPs provide.

With the danger of seaming somewhat pedantic to some readers, a little bit more motivation for use of formal techniques for security protocols is in order here. Arguably, for security systems perfection and assurance of perfection are highly important, since bugs cannot be considered “features”, as is the case in other areas. For a security system one often wants to be provided with guarantees that some expected security properties are met. This can be even more difficult to achieve for security ceremonies, which are more complex, composing protocols, including hidden assumptions and human models. In the case of security protocols one hardly can rely on testing to provide assurance; and experience has shown that protocols that are thoroughly tested in practice for years turn out to contain serious flaws, where a famous example is [26].

This motivates why considerable amounts of research have been put into providing mathematical models and theories for studying security protocols. But more practical are the formal tools that have been built on top of these theories so to have a (semi-)automated way of ensuring security properties. Examples of tools include model checkers like Murphi [29, 40] or FDR [15] which are push-button tools with yes/no answers; or theorem provers like ProVerif [4] for the symbolic (process calculi) approach and CryptoVerif [5] for the computational approach, or Isabell/HOL [32], which often need interaction with expect users but which achieve stronger results than model checkers do.

The psi-calculi is a framework that goes well with the ProVerif and FDR tools which are also based on process calculi. But more than this, psi-calculi have been built (i.e., all the related meta-results have been proven) using the proof assistant Isabell/HOL [30]. Therefore one could say that psi-calculi could be seen as lying at the intersection of the two kinds of tools, making use of the strengths of both.

A downside of such strong formally grounded frameworks is that they are difficult to use by common developers of security protocols and ceremonies. This is where graphical formalisms usually can provide considerable simplifications and hide formal notation, concentrating on the concepts and methods instead. A quite appreciated example of graphical languages grounded in theory can be found in the area of developing reactive or concurrent systems. Here the groundbreaking was achieved through *statecharts* [17] which has become a standard and which have been extended into the more recent *live sequence charts* [18]. A long term goal of the present author is to have a similar graphical language and tool-set for security ceremonies; and this paper tries to identify a path in this direction.

2 Background on psi-calculi

Psi-calculus [3] has been developed as a framework for defining nominal process calculi, like the many variants of the pi-calculus [28]. The psi-calculi framework is based on nominal datatypes, and [3, Sec.2.1] gives a sufficient introduction to nominal sets used in psi-calculi. We will not refer much to nominal datatype i this paper, but refer the reader to the book [36] which contains a thorough treatment

of both the theory behind nominal sets as well as various applications (e.g., see [36, Ch.8] for nominal algebraic datatypes). We expect, though, some familiarity with notions of algebraic datatypes and term algebras.

The psi-calculi framework is parametric; instantiating the parameters accordingly, one obtains an *instance of psi-calculi*, like the pi-calculus, or the cryptographic spi-calculus. These parameters are:

$\mathbf{T}$	terms (data/channels)
$\mathbf{C}$	conditions
$\mathbf{A}$	assertions

which are nominal datatypes not necessarily disjoint; together with the following operators:

$\leftrightarrow : \mathbf{T} \times \mathbf{T} \to \mathbf{C}$	channel equality
$\otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A}$	composition of assertions
$\mathbf{1} \in \mathbf{A}$	minimal assertion
$\vdash \subseteq \mathbf{A} \times \mathbf{C}$	entailment relation

Intuitively, terms can be seen as generated from a signature, as in term algebras; the conditions and assertions can be those from first-order logic; the minimal assertion being top/true, entailment the one from first-order logic, and composition taken as conjunction. We will shortly exemplify how pi-calculus is instantiated in this framework. The operators are usually written infix, i.e.: $M \leftrightarrow N$, $\Psi \otimes \Psi'$, $\Psi \vdash \varphi$.

The above operators need to obey some natural requirements, when instantiated. Channel equality must be symmetric and transitive. The composition of assertions must be associative, commutative, and have $\mathbf{1}$ as unit; moreover, composition must preserve equality of assertions, where two assertions are considered equal iff they entail the same conditions (i.e., for $\Psi, \Psi' \in \mathbf{A}$ we define the equality $\Psi \simeq \Psi'$ iff $\forall \varphi \in \mathbf{C} : \Psi \vdash \varphi \Leftrightarrow \Psi' \vdash \varphi$).

The intuition is that assertions will be used to assert about the environment of the processes. Conditions will be used as guards for guarded (non-deterministic) choices, and are to be tested against the assertion of the environment for entailment. Terms are used to represent complex data communicated through channels, but will also be used to define the channels themselves, which can thus be more than just mere names, as in pi-calculus. The composition of assertions should capture the notion of combining assumptions from several components of the environment.

The syntax for building psi-process is the following (psi-processes are denoted by the $P, Q, \ldots$; terms from $\mathbf{T}$ by $M, N, \ldots$;):

$\mathbf{0}$	Empty/trivial process
$\overline{M}\langle N \rangle.P$	Output
$\underline{M}\langle(\lambda \tilde{x})N\rangle.P$	Input
case $\varphi_1 : P_1, \ldots, \varphi_n : P_n$	Conditional (non-deterministic) choice
$(\nu a)P$	Restriction of names a inside processes P
$P \mid Q$	Parallel composition
$!P$	Replication
$(\!\mid \Psi \mid\!)$	Assertions

The empty process has the same behavior as, and thus can be modeled by, the trivial assignment $(\!\mid \mathbf{1} \mid\!)$.

The input and output processes are as in pi-calculus only that the channel objects M can be arbitrary terms. In the input process the object $(\lambda \tilde{x})N$ is a pattern with the variables $\tilde{x}$ bound in N as well as in the continuation process P. Intuitively, any term message received on M must match the pattern N for some substitution of the variables $\tilde{x}$. The same substitution is used to substitute these variables in P after a successful match. The traditional pi-calculus input $a(x).P$ would be modeled in psi-calculi as $\underline{a}\langle(\lambda x)x\rangle.P$, where the simple names a are the only terms allowed.

The case process behaves like one of the P_i for which the condition φ_i is entailed by the current environment assumption, as defined by the notion of *frame* which we preset later. This notion of frame is familiar from the applied pi-calculus, where it was introduced with the purpose of capturing static information about the environment (or seen in reverse, the frame is the static information that the current process exposes to the environment). A particular use of case is as **case** $\varphi : P$ which can be read as **if** φ **then** P. Another special usage of case is as **case** $\top : P_1, \top : P_2$, where $\Psi \vdash \top$ is a special condition that is entailed by any assertion, like $a \leftrightarrow a$; this use is mimicking the pi-calculus nondeterministic choice $P_1 + P_2$. Restriction, parallel, and replication are the standard constructs of pi-calculus.

Assertions $(\!|\Psi|\!)$ can float freely in a process (i.e., be put in parallel) describing assumptions about the environment. Otherwise, assertions can appear at the end of a sequence of input/output actions, i.e., these are the guarantees that a process provides after it finishes (on the same lines as in assume/guarantee reasoning about programs). Assertions are somehow similar to the active substitutions of the applied pi-calculus, only that assertions do not have computational behavior, but only restrict the behavior of the other constructs by providing their assumptions about the environment.

Example 2.1 (pi-calculus as an instance) *To obtain pi-calculus [28] as an instance of psi-calculus use the following, built over a single set of names $\mathcal{N}$:*

$$\begin{array}{lcl} \mathbf{T} & \triangleq & \mathcal{N} \\ \mathbf{C} & \triangleq & \{a = a \mid a, b \in \mathbf{T}\} \\ \mathbf{A} & \triangleq & \{\mathbf{1}\} \\ \leftrightarrow & \triangleq & = \\ \vdash & \triangleq & \{(\mathbf{1}, a = a) \mid a \in \mathbf{T}\} \end{array}$$

with the trivial definition for the composition operation. The only terms are the channel names $a \in \mathcal{N}$, and there is no other assertion than the unit. The conditions are equality tests for channel names, where the only successful tests are those where the names are equal. Hence, channel comparison is defined as just name equality.

Psi-calculus is given an operational semantics in [3] using labeled transition systems, where the nodes are the process terms and the transitions represent one reduction step, labeled with the action that the process executes. The actions, generally denoted by α, β, represent respectively the input and output constructions, as well as τ the internal synchronization/communication action:

$$\overline{M}\langle(\nu\tilde{a})N\rangle \mid \underline{M}\langle N\rangle \mid \tau$$

Transitions are done in a context, which is represented as an assertion Ψ, capturing assumptions about the environment:

$$\Psi \rhd P \xrightarrow{\alpha} P'$$

Intuitively, the above transition could be read as: The process P can perform an action α in an environment respecting the assumptions in Ψ, after which it would behave like the process P'.

The context assertion is obtained using the notion of *frame* which essentially collects (using the composition operation) the outer-most assertions of a process. The frame $\mathcal{F}(P)$ is defined inductively on the structure of the process as:

$$\begin{array}{l} \mathcal{F}((\!|\Psi|\!)) = \Psi \\ \mathcal{F}(P \mid Q) = \mathcal{F}(P) \otimes \mathcal{F}(Q) \\ \mathcal{F}((\nu a)P) = (\nu a)\mathcal{F}(P) \\ \mathcal{F}(!P) = \mathcal{F}(\textbf{case } \tilde{\varphi} : \tilde{P}) = \mathcal{F}(\overline{M}\langle N\rangle.P) = \mathcal{F}(\underline{M}\langle(\lambda\tilde{x})N\rangle.P) = \mathbf{1} \end{array}$$

Any assertion that occurs under an action prefix or a condition is not visible in the frame.

We give only an exemplification of the transition rules for psi-calculus, and refer to [3, Table 1] for the full definition. The (CASE) rule shows how the conditions are tested against the context assertions. The communication rule (COM) shows how the environment processes executing in parallel contribute their top-most assertions to make the new context assertion for the input-output action of the other parallel processes. In the (com)-rule the assertions Ψ_P and Ψ_Q come from the frames of $\mathscr{F}(P) = (\nu \tilde{b_P})\Psi_P$ respectively $\mathscr{F}(Q) = (\nu \tilde{b_Q})\Psi_Q$.

$$\frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \tilde{\varphi} : \tilde{P} \xrightarrow{\alpha} P'}\ (\text{CASE})$$

$$\frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}\langle(\nu \tilde{a})N\rangle} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\langle N\rangle} Q' \qquad \Psi_Q \otimes \Psi_P \otimes \Psi \vdash M \leftrightarrow K}{\Psi \rhd P \,|\, Q \xrightarrow{\tau} (\nu \tilde{a})(P' \,|\, Q')}\ (\text{COM})$$

There is no transition rule for the assertion process; this is only used in constructing frames. Once an assertion process is reached, the computation stops, and this assertion remains floating among the other parallel processes and will be composed part of the frames, when necessary, like in the case of the communication rule.

3 A psi-calculus instance for actor network procedures

We do not introduce the notation and definitions used in the ANPs of [33] because our aim here is to develop teh ANP ideas in the formal language of psi-calculi. In consequence, we lie to see this section as a formal description of ANPs. The ideas and development from [33] of the ANPs require quite expressive theories which cannot be easily captured with traditional formalisms for security protocols, but which are available in the psi-calculi framework.

There are a few aspects of psi-calculi that offer us the possibility to give semantics to the actor network procedures in this section; and in fact to complex systems like ubiquitous systems where humans are part of the system.

One aspect is the expressiveness of the terms that are allowed to be used for representing messages. This is very liberal in psi-calculi, and thus can easily capture complex structures of messages. Moreover, nominals are allowed in the terms for capturing the important notion of names (like in pi-calculi, or object-oriented languages). Names appear in actor network procedures in three places, as we see further, as names for channels, system configurations, and names of participants in the ceremony.

Another aspect of psi-calculi is the way communication can be done through arbitrarily complex communication terms. This means we are not restricted to just one channel name, but more structure for channels is allowed. We exploit this when modeling the structure of the system configurations and their attached channels and participants. This more complex structure is responsible for the communications in the actor network procedures.

An important aspect of psi-calculi is also the logic that is available through the assertions and the conditions language, and the entailment relation between the two. The liberty that the psi-calculi framework offers for defining the logical part of the calculus allows one to choose the right language for the application purpose. In consequence, depending on the problem one can choose a more expressive logical entailment or one with better computation properties (i.e., feasible for automation). One way of using the assertions and conditions is as in the Hoare-style of reasoning. We may have pre-conditions (using the case construct) and post-conditions (using the trailing assertions) for individual actions as well

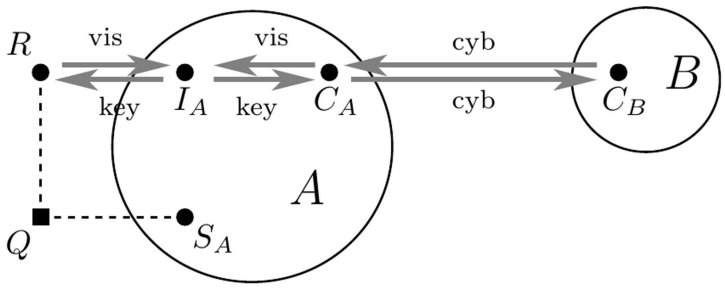

Figure 1: The structure of the configurations, part of the CAP procedure.

as for complex processes. Essentially, psi-calculi allow us to have an assume/guarantee reasoning style using the logical language of our choice and the granularity of the reasoning (i.e., process/action level). Processes are annotated with logical assertions so that an external logical reasoning system can be used on top of the process. But as well, the conditions are logically tested by the process while running, constituting a runtime level reasoning system.

The logical part of the psi-calculi will be exploited to capture the reasoning aspects that actor networks procedures and their PDL logic offers.

3.1 Example of an actor network procedure

We take the *two-factor authentication* example from [33] of the Chip Authentication Program (CAP) procedure. The configuration structure is described in [33, 2.4.2] whereas the runs of the ceremony are described in [33, 3.5.1]. The graphical formalization of this ceremony is given in [33, Fig.3] for the structure and in [33, Fig.7] for the run; we will use the same original figures so to stick with the graphical choices of the authors of [33].

The Figure 1 (taken from [33, Fig.3]) graphically represents the structure of the configurations and the attached communication channels for the actor network procedure that formalizes the CAP two-factor authentication [10]. The structure contains two identity names A (for Alice) and B (for the Bank) which are attached (as subscript) to some of the configurations. The circled areas are not strictly necessary, and become impossible to represent for larger examples; but are good visual aid for examples like this one where they encircle all those configurations corresponding to the respective identity.

The single configuration C_B represents the Bank's computer. Three configurations are under A's control: the computer C_A, the card S_A, and the human representation of Alice I_A. A card reader R is also available, which when coupled with Alice's card form the configuration Q. The human I_A can output through a keyboard channel information to Alice's computer and through another keyboard channel to the card reader. Both the card reader and the computer have visual displays that give information to the human. There are two cyber channels between the two computers; cyber channels are assumed to be untrusted and the information on them should be transmitted encrypted.

The arrows represent channels, and have attached a label denoting the type of the channel. The dark circles and squares represent configurations, where the squares are complex ones containing sub-configurations, whereas the circles are minimal configurations; which are called nodes in [33]. The dashed lines display the containment relation between the configurations.

Based on this structure, runs are drawn (usually one run, the desired/secure run). The Figure 2 (taken from [33, Fig.7]) graphically represents the desired run for the CAP authentication. For a better visual association, the structure of the configurations is displayed at the top, in a more simplistic manner.

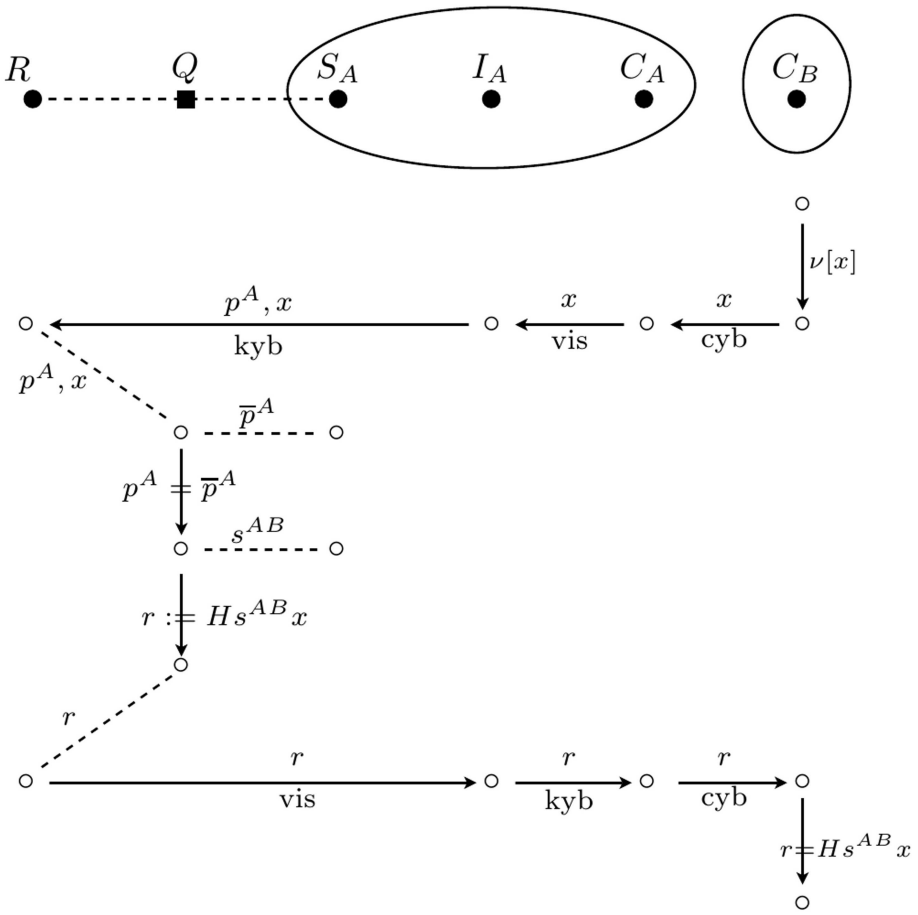

Figure 2: The desired run of the CAP procedure.

The run in Figure 2 shows several aspects of ANPs. Internal actions are drawn vertically, whereas communication between configurations are drawn horizontally. The actions are displayed on the arrows, as well as the channel type that is used for communication. There are actions of generation of fresh content, of transmission of information, of polyadic communication (tuples of messages are sent), tests, and assignments. The dashed lines again are related to configurations made from combination of other configurations, but in a run represent sharing of information.

In this run the Bank generates a fresh value x and sends it to the computer of Alice which communicates it to the human through a visual display channel. The human forwards this value and a password to the card reader through a keyboard input channel. The card reader and the card form a configuration inside which the information sent by the human is shared. This configuration compares the password send by the human to the one stored on the card S_A. If matched then the card gives away a long secret s^{AB}, which is difficult for a human to remember, opposed to a password. The configuration Q hashes this secret and the fresh value into a response which the card reader displays to the human to copy and forward it through the computer of Alice to the Bank's computer. The long secret s^{AB} is shared a priori between the card and the Bank. Therefore the Bank can perform the same calculation to generate a hashed value from this secret and the fresh value and to test it against the received response.

The fresh value is used to ensure that phishing cannot be done through recording just one session. The secret stored on the card is assumed to be a strong secret. The password is used only to make sure

that the right human has made the configuration Q by inserting her card into the card reader. Sending this weak password is done through a physical channel, which is assumed to be harder to attack.

3.2 Encoding

Actor network procedures essentially consist of a structure of configurations together with events/actions causally ordered in the concurrent runs of the procedure/protocol. Configurations are partially ordered by a containment relation, which specifies which sub-configurations form a larger configuration. Configurations have attached channel ends (input and output ends). There is information flow inside configurations; more details will come later in the text.

We therefore, identify one nominal datatype built over a set of *configuration names* $\mathcal{N}_C$. This datatype captures the partial order on the configurations. The terms that we will use are lists of configuration names describing reachability paths based on the parent-child relation between configurations; i.e., each configuration comes with the list of its ancestors.

$$[l_1, \cdots, l_n] \in \mathbf{T} \text{ for } l_i \in \mathcal{N}_C.$$

The list terms may also contain variables, and the names in the list may also be hidden behind a name binding restriction operation $(\nu \cdot)$.

Channels and channel types do not bare much distinction in [33]. In consequence we will treat them in this paper as one and the same *channel name*. If proper channel types are needed (like stating what kind of messages are allowed to be communicated) we could turn to the work on typed psi-calculi of [23] which extends the classic works on typed pi-calculus [34, 39] or on the distributed pi-calculus [22, 21] where types capture resources. More complex types of channels, like the ones that [33] talks about, could be captured with complex processes that are processing the messages received, before forwarding them to the intended recipient. This is how channels like a *random noise binary channel* would be described, or a *lossy channel*. Moreover, we do not restrict the formalism and do not assume that only one channel of one type exists between two configurations, as is done in [33].

In consequence, the nominal datatype from before is enriched with a set of *channel names* $\mathcal{N}_A$ which are paired with the list terms. This describes how a channel name is attached to the configuration described by the list term. In consequence a channel is represented by a term:

$$[l_1, \cdots, l_n]c \in \mathbf{T} \text{ for } c \in \mathcal{N}_A,$$

where the channel name is c and the list $[l_1, \cdots, l_n]$ denotes the configuration (and it's ancestor configurations) to which c is attached.

Communication in psi-calculus on such a channel is defined with the psi-calculus process syntax from the previous section, e.g.:

$$\overline{[l_1, \cdots, l_n]c}\langle N \rangle \quad \text{in ANP notation would be} \quad \overline{c}\langle [N]_{[l_1, \cdots, l_n]} \rangle$$

where $N \in \mathbf{T}$ is usually a message term. Intuitively, this says that there is an output (sending) of the message N on the channel c in the configuration $[l_1, \cdots, l_n]$.

We allow message terms to be constructed from some arbitrary signature, as it is allowed in the actor network procedures, and supported by the psi-calculi framework. For the purposes of this paper, the signature for building messages N is left open, and is unimportant for the developments that we do further. For a specific application to a security ceremony or protocol, the designer decides on the message terms needed in the procedure. In this way we allow one to specify the minimal signature for

message terms as needed for the specific protocol. Therefore the computation needed to analyze the specific protocol is dependent on the messages being sent part of the protocol.

One more nominal set $\mathcal{N}_I$ keeps track of the *identities* involved in the ceremony. Identities are associated to configurations through a partial function, describing which identity controls which configuration; some configurations may be uncontrolled (thus the partiality of the function). We use a pairing construct to form configurations which are controlled by some principal:

$$i[l_1,\cdots,l_n] \in \mathbf{T} \text{ for } i \in \mathcal{N}_I.$$

Channels can be attached to both uncontrolled configurations, like we did before, as well as to controlled configurations:

$$i[l_1,\cdots,l_n]c \in \mathbf{T} \text{ for } i \in \mathcal{N}_I, l_i \in \mathcal{N}_A, c \in \mathcal{N}_C.$$

This example of term involves all three nominal sets, and is arguably the most complex form of terms we will use for communications in an ANP. More complication would come only from the specific term algebra of the messages that a designer chooses for a specific protocol.

In the actor network procedures, the communication operations which are not controlled, i.e.,

$$\overline{[l_1,\cdots,l_n]c}\langle N\rangle \quad \text{as Output, or} \qquad \underline{[l_1,\cdots,l_n]c}\langle(\nu\tilde{a})N\rangle \quad \text{as Input}$$

are called *events*, whereas the controlled ones, like

$$\overline{i[l_1,\cdots,l_n]c}\langle N\rangle \quad \text{or} \qquad \underline{i[l_1,\cdots,l_n]c}\langle(\nu\tilde{a})N\rangle,$$

are called *actions*. But there is no essential difference, and thus the graphical notation does not make a distinction between the two.

The structure of the actor network procedure is intended to capture how information is shared within the sub-configurations. In particular, information could flow from a main configuration to its sub-configurations when received on a channel attached to the main configuration. Opposite, information could flow from a sub-configuration to its parents (and ancestors) configurations, and be sent out through channels attached to an ancestor, and not to the originator configuration. This form of information sharing is rather open and liberal in ANPs, which would leave room for a lot of hidden flow of information.

This means that there is not much control on the sharing of information in ANPs. It may be that the parent configuration wants to share some information only with part of its sub-configurations. In consequence, here we allow for more control on the information flow so a communication action can specify explicitly to which sub-configuration it wants to communicate. Moreover, we allow for hiding of the internal structure of configurations. When hiding is used then sharing can be implemented only internally, by first communicating with the main (public) configuration, which in turn decides to which sub-configurations to forward the message (possibly changed).

Until now we have encoded the communication structure of an actor network procedure through the terms $\mathbf{T}$ of psi-calculus. The approach that we took above is inspired by the nested distributed pi-calculus of [24], where locations can be nested (i.e., a location can have sub-locations). Depending on the list terms that one defines, different relations between the configurations can be obtained. For the ANPs in particular, we are aiming for a partial order relation.

But the formalism for ANPs should subsume existing formalisms for security protocols and communicating processes. Indeed, if the lists defined above are always empty, and only channel names are used, then we obtain the communication mechanism of pi-calculus. Assume that no identities are present in the terms. The distributed pi-calculus is then obtained when working only with singleton lists, i.e., a

flat structure of the locations (or configurations). Ambient calculi [7] or bigraphs [27] are obtained by making a tree-like structure between the configurations.

Internal sharing can be captured using local communication, hidden from outsiders. The same approach we use to model local actions, like assignment or generation of fresh names as in the CAP example. Sending a fresh value on the cyber channel in the CAP example would be encoded as:

$$(\nu fr)\,\overline{B[l_C]cyb}\langle fr\rangle.\mathbf{0}$$

For ease of notation, henceforth we will not add the empty process at the end. To encode a local/internal action we use a private channel on which the configuration communicates with itself, as:

$$(\nu loc)(\overline{B[l_C]loc}\langle fr\rangle \mid \underline{B[l_C]loc}\langle(\lambda x)x\rangle).$$

Example 3.1 *The local generation of the fresh value fr by Bank's computer $B[l_C]$, which is then communicated on the cyber channel is thus modeled as:*

$$(\nu loc)((\nu fr)\overline{B[l_C]loc}\langle fr\rangle \mid \underline{B[l_C]loc}\langle(\lambda x)x\rangle.\overline{B[l_C]cyb}\langle x\rangle).$$

The restriction operator ν applied to the name loc makes this communication channel visible only inside the scope of ν, whereas the restriction of the name fr models the uniqueness, thus freshness.

Up to now we have made use only of the **T** nominal datatype of the psi-calculi, not needing the logical part offered by the assertions **A** and conditions **C**. The terms we described until now, part of **T**, are used to capture the complex communication structure of ANP. This was used in conjunction with the process syntax for input/output, parallel composition and name restriction.

The **case** process of psi-calculus is powerful, offering both non-determinism as well as conditionals. Actor network procedures do not involve non-determinism, the same as the spi- and applied pi-calculus. These require only conditional constructs. This is natural when thinking that these are formalisms for describing security/communication protocols, i.e., deterministic runs of such protocols. But having the non-deterministic choice possibility in psi-calculi opens up for more modeling opportunities, like when wanting to refine the model of the human into a probabilistic one, involving probabilistic choices.

We have seen the necessity for the conditional in the CAP example, where terms were tested for equality. In consequence, we will include as conditions in **C**, tests for equality for any two nominal terms from **T**:

$$M = N \in \mathbf{C} \;\text{ for any }\; M, N \in \mathbf{T}.$$

The entailment relation defines when two terms are equal wrt. some assertion:

$$\Psi \vdash M = N \;\text{ iff }\; \vdash_\Sigma M = N$$

where Σ is the signature over which the message terms are built, and $\vdash_\Sigma$ is the equational logic entailment relation wrt. the signature Σ. In other words, $M = N$ is decided only looking at the terms, using syntactic unification in the term algebra described by the signature Σ. In many cases equations are defined for such a term algebra, which need to be considered when deciding the equality of two terms. This would then involve working modulo these equations; we use the same notation $\vdash_\Sigma$ for the case when equations are part of the definition of the algebra of terms too.

Testing for equality of message terms does not depend on the assertions. Nevertheless, we will use assertions to model the partially ordered runs that actor network procedures use. This way of capturing true concurrency models (like the pomsets [13, 37] used by ANPs) in psi-calculi is inspired by the recent [31]. Actor network procedures is among the few formalisms that acknowledges the need for true

concurrency [42] when modeling protocols, instead of interleaving concurrency or linear runs as most process algebras approaches do. Actor network procedures describe a run to be a *pomset*, i.e., a partially ordered multiset. Knowing that a linear run (word, string) is a totally ordered multiset (because symbols may appear multiple times in the string), then a pomset is a partially ordered run, i.e., a run where some of the actions are concurrent, not all being linearly ordered. It is natural to model a run of a protocol or ceremony as a partially ordered run because there are several parties involved, often distributed, thus executing actions/communication in parallel, concurrently. A single participant may be deterministic and sequential, thus exhibiting a linear run. But put together several participants exhibit a partially ordered run of the whole ceremony. More order constraints can come from the communications, where e.g., input actions depend on (i.e., must come after) the respective output actions.

Actor network procedures are declarative when defining their runs, in the same style as true concurrency models like event structures [41] or pomsets, or as languages are, as opposed to automata or process algebras which describe their runs in an operational manner. Psi-calculi allow also a declarative style of defining dependencies between actions by using the logic captured in the entailment relation and the assertions. In this way we have the full descriptive power of true concurrency models so to capture conjunctive (or disjunctive) dependencies as is the case with pomsets. This cannot be captured only through the sequence operation of the psi-processes.

We define assertions $\mathbf{A}$ to contain sets of communication terms, e.g.:

$$\{\underline{i[l_1,\ldots,l_k]c}\langle N\rangle, \overline{i[l_1,\ldots,l_k]c}\langle N\rangle,\ldots\} \in \mathbf{A} \text{ with } i[l_1,\ldots,l_k]c \in \mathbf{T} \text{ and } N \in \mathbf{T}.$$

In consequence, conditions are also enriched with such actions by combining them with conjunction. The entailment is defined to treat conjunction appropriately, as in classical logics. For the new kind of conditions the entailment is just set-containment:

$$\Psi \vdash \underline{i[l_1,\ldots,l_k]c}\langle N\rangle \text{ iff } \underline{i[l_1,\ldots,l_k]c}\langle N\rangle \in \Psi.$$

With the conditions and assertions in place we can capture orders on the communication actions in the form of pomsets as follows. Each action is conditioned by a set of other actions on which it depends. In this way the action cannot be performed until the condition is met, i.e., all the actions on which it depends have been done. Thus,

$$\textbf{case } \varphi : \underline{i[l_1,l_2]c}\langle N\rangle.P \text{ with } \varphi = \{\overline{j[l_1]d}\langle N'\rangle, \underline{i[l_1,l_2]b}\langle N''\rangle\}$$

describes the fact that action $\underline{i[l_1,l_2]c}\langle N\rangle$ must come after the two actions from the condition have been executed. The knowledge that an action has been executed is gathered in the context assertion through assertion processes which are left behind after each execution of an action. This is easily done by changing the continuation of an action:

$$\underline{i[l_1,l_2]c}\langle N\rangle.P \text{ becomes } \underline{i[l_1,l_2]c}\langle N\rangle.(P \mid (\!|\underline{i[l_1,l_2]c}\langle N\rangle|\!)).$$

After the action is performed, the trailing assertion will become available to both the continuation and the other parallel processes, as part of the context assertion collected by the frame of the parallel process.

The actor network procedure formalism uses the PDL (Procedure Description Logic) to reason about runs. PDL[1] uses two kinds of basic formulas: one states that an action has been executed; and another

[1]PDL for actor network procedures should not be confused with Propositional Dynamic Logic [20] (usually abbreviated PDL, or DL for the higher order case) used for reasoning about programs. On the other hand, propositional dynamic logic is a modal logic that reasons about actions in general, and could also be used for reasoning about ANPs once the special basic formulas and actions are set, as done for the ANPs. In fact, the reasoning style of PDL for ANPs resembles much the past temporal logic style of reasoning, and temporal reasoning can be done with propositional dynamic logic too.

states that some action has been executed before another action. Above, the assertions capture only the first kind of PDL basic formula. We now add another assertion that stands for the second kind of PDL formula. This second kind of assertions capture a whole pomset in the assertions only. This pomset is available to the process for inspection, during the execution, and it captures the partial run so far. It is like a history in Hoare-style reasoning, only that in our case it is a partially ordered history.

We thus add to the assertion terms, dependency pairs of actions, giving the possibility to describe partial orders of actions as assertions:

$$\underline{i[l_1,\ldots,l_k]c}\langle N\rangle \leq \overline{j[l_1,\ldots,l_k]d}\langle N'\rangle \in \mathbf{A}$$

signifying that the right-hand action depends on the left-hand action. If, moreover, both these actions are part of the assertions set then we conclude that the left action happened before the right action.

A question to ask is how do such dependency pairs get into the assertion set. Trailing assertion processes would be used, the same as we did earlier, only that more care needs to be taken when defining the assertion composition operation. We are not just using set union, but for building dependency pairs we must also achieve the transitivity property of the partial order relation we want to maintain.

Example 3.2 *For the CAP run in Figure 2 the execution of the process would reach a point (e.g., upper left-most corner) where the environment assertion* Ψ *would contain a dependency pair*

$$\overline{A[l_C]cyb}\langle fr\rangle \leq \underline{A[l_I]kyb}\langle(p^A, fr)\rangle,$$

saying that the action of Alice of typing at the keyboard of the password and the fresh value is dependent, thus should come after, the computer of Alice having received the fresh value.

We have thus covered all aspects of the actor network procedures graphical formalism through the psi-calculus operational formalism. The structure of the configurations has been captured through the nominal datatypes, and the message terms have been treated the same as in ANPs. The definition of the pomset runs of a ANP was done by making use of the encoding of the dependencies between the actions using the assertions and conditions. Communication is done through channels attached to configurations, and internal actions are modeled also as communications but on private channels.

We have thus seen use of all psi-constructs for building processes: input/output used for communication and simulation of other actions like assignment; the **case** for modeling conditionals (and Hoare-style pre-conditions); restriction of names for modeling private communications and fresh values; parallel composition for putting several identities in the protocol to run together; and assertion processes for capturing the Hoare-style guarantees. The *replication* construct has not been used; but it is essentially useful when needing to model ceremonies that can run through several sessions.

4 Conclusions and outlook

There are many possibilities that psi-calculi open up for modeling ubiquitous systems and security ceremonies, where humans are part of the system and are intended to be modeled in a more faithful way. The work we undertook in this paper shows that the psi-calculi framework is expressive enough to capture faithfully complex formalisms like the actor network procedures, and even with some generalizations thereof. We could easily capture concurrent computations in psi-calculi, besides sequential ones.

The logic that psi-calculi offers is opened to be tailored to the application needs. In the case of ANPs we could use it to capture the Hoare-style reasoning that ANPs use. Such a reasoning is essential for security ceremonies where the assumptions should be made explicit, opposed to what usually is done with formalisms for security protocols where many assumptions are left underspecified.

The term construction mechanisms are also rather liberal in psi-calculi. This offers the possibility to define with any degree of detail needed, the message terms exchanged in the ceremony. At the same time we are not constrained when defining communication terms. This allows for modeling a great wealth of communication mechanisms. We have exploited this second freedom in the construction of the nominal terms representing the complex communication structure of ANPs.

For ceremonies in particular, we are interested in more faithful modeling of the human nodes where we would like to either leave room for errors, i.e., using non-determinism, or we would like to integrate a statistical model of the human, using probabilistic choices. These go beyond what current actor network procedures allow. But psi-calculi can accommodate probabilistic models, e.g., by going through CC-pi [6, 9] which has already been treated as a psi-calculus. But the work on probabilities and psi-calculus is not yet mature and still needs more investigation.

An interesting future direction for a graphical formalism line the ANPs for modeling ceremonies is the notion of action refinement [14]. This is a technique for building (and working with) models in an incremental way, starting from an abstraction and refining actions into more concrete implementations. Action refinement is well behaved for true concurrency models and their equivalences like history preserving bisimulation. But it is not studied how to do action refined for graphical languages (except for the statecharts [17]), and not for ANPs either. Refinement for ANPs would allow a ceremony designer to refine abstract models, both the configuration structures and the runs. By refining, one can expand single actions into more complex runs, or can expand one configuration with sub-configurations.

Action refinement is a technique for building models compositionally, in a top-down manner, whereas process algebras have a compositional approach where they build a model from components plugged together using operators like choice, sequential, or parallel composition. Action refinement can work well in combination with the compositional approach of process algebras; and we encourage this combination.

The structure of the configurations in an ANP is static. It is not the case that during the execution of the ceremony some configurations are broken, disappear, or loose some of their sub-configurations. But in the psi-calculi one can easily model a dynamic structure, similar to how the ambients are dynamic in ambient calculus [7], or how communication changes locations in distributed pi-calculus [21]. The recent bigraphs formalism [27, 16] is especially focusing on how the structure of the system changes; the execution is defined in terms of change of structure (and interaction links). We see great possibility for investigating change of structure in ANPs, starting from the encoding we have given in this paper, and from the investigations of the above formalisms wrt. the psi-calculi framework.

References

[1] Martín Abadi & Cédric Fournet (2001): *Mobile values, new names, and secure communication*. In Chris Hankin & Dave Schmidt, editors: *POPL*, ACM, pp. 104–115. Available at `http://doi.acm.org/10.1145/360204.360213`.

[2] Martín Abadi & Andrew D. Gordon (1999): *A Calculus for Cryptographic Protocols: The spi Calculus*. *Inf. Comput.* 148(1), pp. 1–70. Available at `http://dx.doi.org/10.1006/inco.1998.2740`.

[3] Jesper Bengtson, Magnus Johansson, Joachim Parrow & Björn Victor (2011): *Psi-calculi: a framework for mobile processes with nominal data and logic*. *Logical Methods in Computer Science* 7(1). Available at `http://dx.doi.org/10.2168/LMCS-7(1:11)2011`.

[4] Bruno Blanchet (2004): *Automatic Proof of Strong Secrecy for Security Protocols*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 86–102. Available at `http://doi.ieeecomputersociety.org/10.1109/SECPRI.2004.1301317`.

[5] Bruno Blanchet (2008): *A Computationally Sound Mechanized Prover for Security Protocols*. *IEEE Trans. Dependable Sec. Comput.* 5(4), pp. 193–207. Available at http://dx.doi.org/10.1109/TDSC.2007.1005.

[6] Maria Grazia Buscemi & Ugo Montanari (2007): *CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements*. In Rocco De Nicola, editor: *16th European Symposium on Programming Programming Languages and Systems (ESOP'07)*, *LNCS* 4421, Springer, pp. 18–32. Available at http://dx.doi.org/10.1007/978-3-540-71316-6_3.

[7] Luca Cardelli & Andrew D. Gordon (1998): *Mobile Ambients*. In M. Nivat, editor: *Foundations of Software Science and Computation Structure, FoSSaCS'98*, *Lecture Notes in Computer Science* 1378, Springer, pp. 140–155. Available at http://dx.doi.org/10.1007/BFb0053547.

[8] Anupam Datta, John C. Mitchell, Arnab Roy & Stephan Hyeonjun Stiller (2011): *Protocol Composition Logic*. In V. Cortier & S. Kremer, editors: *Formal Models and Techniques for Analyzing Security Protocols*, IOS Press.

[9] Rocco De Nicola, Gian Luigi Ferrari, Ugo Montanari, Rosario Pugliese & Emilio Tuosto (2005): *A Process Calculus for QoS-Aware Applications*. In Jean-Marie Jacquet & Gian Pietro Picco, editors: *7th International Conference on Coordination Models and Languages*, *LNCS* 3454, Springer, pp. 33–48. Available at http://dx.doi.org/10.1007/11417019_3.

[10] Saar Drimer, Steven J. Murdoch & Ross J. Anderson (2009): *Optimised to Fail: Card Readers for Online Banking*. In Roger Dingledine & Philippe Golle, editors: *Financial Cryptography and Data Security*, *LNCS* 5628, Springer, pp. 184–200. Available at http://dx.doi.org/10.1007/978-3-642-03549-4_11.

[11] Nancy A. Durgin, John C. Mitchell & Dusko Pavlovic (2003): *A Compositional Logic for Proving Security Properties of Protocols*. *Journal of Computer Security* 11(4), pp. 677–722. Available at http://iospress.metapress.com/content/lgf63yyhl3ajnddt/.

[12] Carl Ellison (2007): *Ceremony Design and Analysis*. Cryptology ePrint Archive, Report 2007/399. Available at http://eprint.iacr.org/2007/399.

[13] Jay L. Gischer (1984): *Partial Orders and the Axiomatic Theory of Shuffle*. Ph.D. thesis, CS, Stanford University.

[14] Rob J. van Glabbeek & Ursula Goltz (2001): *Refinement of actions and equivalence notions for concurrent systems*. *Acta Informatica* 37(4/5), pp. 229–327. Available at http://link.springer.de/link/service/journals/00236/bibs/1037004/10370229.htm.

[15] Michael Goldsmith, Gavin Lowe, Bill Roscoe, Peter Ryan & Steve Schneider (2000): *Modelling and analysis of security protocols*. Pearson Education.

[16] Davide Grohmann (2008): *Security, Cryptography and Directed Bigraphs*. In: *4th International Conference on Graph Transformations (ICGT'08)*, *LNCS* 5214, Springer-Verlag, pp. 487–489. Available at http://dx.doi.org/10.1007/978-3-540-87405-8_41.

[17] David Harel (1987): *Statecharts: A Visual Formulation for Complex Systems*. *Science of Computer Programmings* 8(3), pp. 231–274. Available at http://dx.doi.org/10.1016/0167-6423(87)90035-9.

[18] David Harel & Rami Marelly (2003): *Come, Let's Play – Scenario-Based Programming Using LSCs and the Play-Engine*. Springer. Available at http://www.springer.com/computer/programming/book/978-3-540-00787-6.

[19] David Harel & Amnon Naamad (1996): *The STATEMATE Semantics of Statecharts*. *ACM Trans. Softw. Eng. Methodol.* 5(4), pp. 293–333. Available at http://doi.acm.org/10.1145/235321.235322.

[20] David Harel, Jerzy Tiuryn & Dexter Kozen (2000): *Dynamic Logic*. MIT Press, Cambridge, MA, USA.

[21] Matthew Hennessy (2007): *A Distributed Pi-Calculus*. Cambridge Univ. Press.

[22] Matthew Hennessy & James Riely (2002): *Resource Access Control in Systems of Mobile Agents*. *Inf. Comput.* 173(1), pp. 82–120. Available at http://dx.doi.org/10.1006/inco.2001.3089.

[23] Hans Hüttel (2011): *Typed psi-calculi*. In Joost-Pieter Katoen & Barbara König, editors: *CONCUR, Lecture Notes in Computer Science* 6901, Springer, pp. 265–279. Available at http://dx.doi.org/10.1007/978-3-642-23217-6_18.

[24] Hans Hüttel (2013): *On Representing Located Process Calculi in the psi-calculus*. personal communication.

[25] Bruno Latour (2005): *Reassembling the Social – An Introduction to Actor-Network-Theory*. Oxford Univ. Press.

[26] Gavin Lowe (1996): *Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR*. *Software - Concepts and Tools* 17(3), pp. 93–102.

[27] Robin Milner (2009): *The Space and Motion of Communicating Agents*. Cambridge Univ.Press.

[28] Robin Milner, Joachim Parrow & David Walker (1992): *A Calculus of Mobile Processes, I-II*. *Information and Computation* 100(1), pp. 1–77, doi:dx.doi.org/10.1016/0890-5401(92)90008-4.

[29] John C. Mitchell, Mark Mitchell & Ulrich Stern (1997): *Automated analysis of cryptographic protocols using Murphi*. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 141–151. Available at http://doi.ieeecomputersociety.org/10.1109/SECPRI.1997.601329.

[30] Tobias Nipkow, Lawrence C. Paulson & Markus Wenzel (2002): *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*. *Lecture Notes in Computer Science* 2283, Springer. Available at http://dx.doi.org/10.1007/3-540-45949-9.

[31] Håkon Norman (2013): *Event Structures as Psi-calculi*. In Tarmo Uustalu & Juri Vain, editors: *25th Nordic Workshop on Programming Theory (NWPT13)*.

[32] Lawrence C. Paulson (1998): *The Inductive Approach to Verifying Cryptographic Protocols*. *Journal of Computer Security* 6(1-2), pp. 85–128. Available at http://iospress.metapress.com/content/5wlu8p2am1du051d/.

[33] Dusko Pavlovic & Catherine Meadows (2011): *Actor-network procedures: Modeling multi-factor authentication, device pairing, social interactions*. *arXiv.org*. Available at http://arxiv.org/abs/1106.0706.

[34] Benjamin C. Pierce & Davide Sangiorgi (1996): *Typing and Subtyping for Mobile Processes*. *Mathematical Structures in Computer Science* 6(5), pp. 409–453.

[35] Wolter Pieters (2011): *Representing humans in system security models: An actor-network approach*. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 2(1), pp. 75–92.

[36] Andrew M. Pitts (2013): *Nominal Sets: Names and Symmetry in Computer Science*. *Cambridge Tracts in Theoretical Computer Science* 57, Cambridge Univ. Press.

[37] Vaughan R. Pratt (1986): *Modeling Concurrency with Partial Orders*. *J. Parallel Programming* 15(1), pp. 33–71. Available at http://dx.doi.org/10.1007/BF01379149.

[38] Kenneth Radke, Colin Boyd, Juan Manuel González Nieto & Margot Brereton (2011): *Ceremony Analysis: Strengths and Weaknesses*. In: *26th IFIP-TC-11 International Information Security Conference (SEC), IFIP Advances in Information and Communication Technology* 354, Springer, pp. 104–115. Available at http://dx.doi.org/10.1007/978-3-642-21424-0_9.

[39] Davide Sangiorgi & David Walker (2001): *The π-Calculus: a Theory of Mobile Processes*. Cambridge Univ. Press.

[40] Ulrich Stern & David L. Dill (1997): *Parallelizing the Murphi Verifier*. In Orna Grumberg, editor: *9th International Conference on Computer Aided Verification (CAV), LNCS* 1254, Springer, pp. 256–278. Available at http://dx.doi.org/10.1007/3-540-63166-6_26.

[41] Glynn Winskel (1986): *Event Structures*. In: *Advances in Petri Nets, LNCS* 255, Springer, pp. 325–392. Available at http://dx.doi.org/10.1007/3-540-17906-2_31.

[42] Glynn Winskel & Mogens Nielsen (1995): *Models for Concurrency*. In Samson Abramski, Dov M. Gabbay & Tom S.E. Maibaum, editors: *Handbook of Logic in Computer Science – vol 4 – Semantic Modelling*, Oxford University Press, pp. 1–148.

A Graphical Adversarial Risk Analysis Model for Oil and Gas Drilling Cybersecurity

Aitor Couce Vieira
Secure-NOK AS
Stavanger, Norway
aitorcouce@securenok.com

Siv Hilde Houmb
Secure-NOK AS
Stavanger, Norway
sivhoumb@securenok.com

David Rios Insua
Royal Academy of Sciences
Madrid, Spain
david.rios@urjc.es

Oil and gas drilling is based, increasingly, on operational technology, whose cybersecurity is complicated by several challenges. We propose a graphical model for cybersecurity risk assessment based on Adversarial Risk Analysis to face those challenges. We also provide an example of the model in the context of an offshore drilling rig. The proposed model provides a more formal and comprehensive analysis of risks, still using the standard business language based on decisions, risks, and value.

1 Introduction

Operational technology (OT) refers to "hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events in the enterprise" [20]. It includes technologies such as SCADA systems. Implementing OT and information technology (IT) typically lead to considerable improvements in industrial and business activities, through facilitating the mechanization, automation, and relocation of activities in remote control centers. These changes usually improve the safety of personnel, and both the cost-efficiency and overall effectiveness of operations.

The oil and gas industry (O&G) is increasingly adopting OT solutions, in particular offshore drilling, through drilling control systems (drilling CS) and automation, which have been key innovations over the last few years. The potential of OT is particularly relevant for these activities: centralizing decision-making and supervisory activities at safer places with more and better information; substituting manual mechanical activities by automation; improving data through better and near real-time sensors; and optimizing drilling processes. In turn, they will reduce rig crew and dangerous operations, and improve efficiency in operations, reducing operating costs (typically of about $300,000 per day).

Since many of the involved OT employed in O&G are currently computerized, they have become a major potential target for cyber attacks [38], given their economical relevance, with large stakes at play. Indeed, we may face the actual loss of large oil reserves because of delayed maneuvers, the death of platform personnel, or potential large spills with major environmental impact with potentially catastrophic consequences. Moreover, it is expected that security attacks will soon target several production installations simultaneously with the purpose of sabotaging production, possibly taking advantage of extreme weather events, and attacks oriented towards manipulating or obtaining data or information. Cybersecurity poses several challenges, which are enhanced in the context of operational technology. Such challenges are sketched in the following section.

1.1 Cybersecurity Challenges in Operational Technology

Technical vulnerabilities in operational technology encompass most of those related with IT vulnerabilities [7], complex software [5], and integration with external networks [17]. There are also and specific

Pre-proceedings of
The International Workshop on Graphical Models for Security 2014

OT vulnerabilities [42, 6]. However, OT has also strengths in comparison with typical IT systems employing simpler network dynamics.

Sound organizational cybersecurity is even more important with OT given the risks that these systems bring in. Uncertainties are considerable in both economical and technical sense [2]. Therefore better data about intrusion attempts are required for improving cybersecurity [32], although gathering them is difficult since organizations are reluctant about disclosing such information [39].

More formal approaches to controls and measures are needed to deal with advanced threat agents such as assessing their attack patterns and behavior [19] or implementing intelligent sensor and control algorithms [10]. An additional problem is that metrics used by technical cybersecurity to evaluate risks usually tell little to those evaluating or making-decisions at the organizational cybersecurity level. Understanding the consequences of a cyber attack to an OT system is difficult. They could lead to production losses or the inability to control a plant, multimillion financial losses, and even impact stock prices [7]. One of the key problems for understanding such consequences is that OT systems are also cyber-physical systems (CPS) encompassing both computational and complex physical elements [40].

Risk management is also difficult in this context [31]. Even risk standards differ on how to interpret risk: some of them assess the probabilities of risk, others focus on the vulnerability component [19]. Standards also tend to present oversimplifications that might alter the optimal decision or a proper understanding of the problem, such as the well-known shortcomings of the widely employed risk matrices [12].

Cyber attacks are the continuation of physical attacks by digital means. They are less risky, cheaper, easier to replicate and coordinate, unconstrained by distance [8], and they could be oriented towards causing high impact consequences [5]. It is also difficult to measure data related with attacks such as their rate and severity, or the cost of recovery [2]. Examples include Stuxnet [6], Shamoon [6], and others [10] . Non targeted attacks could be a problem also[9].

Several kinds of highly skilled menaces of different nature (e.g., military, hacktivists, criminal organizations, insiders or even malware agents) can be found in the cyber environment [5], all of them motivated and aware of the possibilities offered by OT [7]. Indeed, the concept Advanced Persistent Threat (APT) has arisen to name some of the threats [26]. The diversity of menaces could be classified according their attitude, skill and time constraints [13], or by their ability to exploit, discover or even create vulnerabilities on the system [5]. Consequently, a sound way to face them is profiling [3] and treating [24] them as adversarial actors.

1.2 Related Work Addressing the Complexities of Cybersecurity Challenges

Several approaches have been proposed to model attackers and attacks, including stochastic modelling [30, 36], attack graph models [22] and attack trees [28], models of directed and intelligent attacks [39]; models based on the kill chain attack phases [19], models of APT attack phases [26], or even frameworks incorporating some aspects of intentionality or a more comprehensive approach to risk such as CORAS [27] or ADVISE [11].

Game theory has provided insights concerning the behavior of several types of attackers – such as cyber criminal APTs – and how to deal with them. The concept of incentives can unify a large variety of agent intents, whereas the concept of utility can integrate incentives and costs in such a way that the agent objectives can be modeled in practice [25]. Important insights from game theory are that the defender with lowest protection level tends to be a target for rational attackers [21], that defenders tend to under-invest in cybersecurity [1], and that the attacker's target selection is costly and hard, and thus it needs to be carefully carried on [15]. In addition to such general findings, some game-theoretic models

exist for cybersecurity or are applicable to it, modelling static and dynamic games in all information contexts [35]. However, game-theoretic models have their limitations [18, 35] such as limited data, the difficulty to identify the end goal of the attacker, the existence of a dynamic and continuous context, and that they are not scalable to the complexity of real cybersecurity problems in consideration. Moreover, from the conceptual point of view, they require common knowledge assumptions that are not tenable in this type of applications.

Additionally, several Bayesian models have been proposed for cybersecurity risk management such as a model for network security risk analysis [41]; a model representing nodes as events and arcs as successful attacks [13]; a dynamic Bayesian model for continuously measuring network security [16]; a model for Security Risk Management incorporating attacker capabilities and behavior [14]: or models for intrusion detection systems (IDS) [4]. However, these models require forecasting attack behavior which is hard to come by.

Adversarial Risk Analysis (ARA) [34] combine ideas from Risk Analysis, Decision Analysis, Game-Theory, and Bayesian Networks to help characterizing the motivations and decisions of the attackers. ARA is emerging as a main methodological development in this area [29], providing a powerful framework to model risk analysis situations with adversaries ready to increase our threats. Applications in physical security may be seen in [37].

1.3 Our Proposal

The challenges that face OT, cybersecurity and the O&G sector create a need of a practical, yet rigorous approach, to deal with them. Work related with such challenges provides interesting insights and tools for specific issues. However, more formal but understandable tools are needed to deal with such problems from a general point of view, without oversimplifying the complexity underlying the problem. We propose a model for cybersecurity risk decisions based on ARA, taking into account the attacker behavior. Additionally, an application of the model in drilling cybersecurity is presented, tailored to decision problems that may arise in offshore rigs employing drilling CS.

2 Model

2.1 Introduction to Adversarial Risk Analysis

ARA aims at providing one-sided prescriptive support to one of the intervening agents, the Defender (she), based on a subjective expected utility model, treating the decisions of the Attacker (he) as uncertainties. In order to predict the Attacker's actions, the Defender models her decision problem and tries to assess her probabilities and utilities but also those of the Attacker, assuming that the adversary is an expected utility maximizer. Since she typically has uncertainty about those, she models it through random probabilities and uncertainties. She propagates such uncertainty to obtain the Attacker's optimal random attack, which she then uses to find her optimal defense.

ARA enriches risk analysis in several ways. While traditional approaches provide information about risk to decision-making, ARA integrates decision-making within risk analysis. ARA assess intentionality thoroughly, enabling the anticipation and even the manipulation of the Attacker decisions. ARA incorporates stronger statistical and mathematical tools to risk analysis that permit a more formal approach of other elements involved in the risk analysis. It improves utility treatment and evaluation. Finally, an ARA graphical model improves the understandability of complex cases, through visualizing the causal relations between nodes.

The main structuring and graphical tool for decision problems are Multi-Agent Influence Diagrams (MAID), a generalization of Bayesian networks. ARA is a decision methodology derived from Influence Diagrams, and it could be structured with the following basic elements:

- *Decisions or Actions*. Set of alternatives which can be implemented by the decision makers. They represent what one can do. They are characterized as decision nodes (rectangles).
- *Uncertain States*. Set of uncontrollable scenarios. They represent what could happen. They are characterized as uncertainty nodes (ovals).
- *Utility and Value*. Set of preferences over the consequences. They represent how the previous elements would affect the agents. They are characterized as value nodes (rhombi).
- *Agents*. Set of people involved in the decision problem: decision makers, experts and affected people. In this context, there are several agents with opposed interests. They are represented through different colors.

We describe now the basic MAID that may serve as a template for cybersecurity problems in O&G drilling CS, developed using GeNIe [23].

2.2 Graphical Model

Our model captures the Defender cybersecurity main decisions prior to an attack perpetrated by an APT, which is strongly "business-oriented". Such cyber criminal organization behavior suits utility-maximizing analysis, as it pursues monetary gains. A sabotage could also be performed by this type of agents, and they could be hired to make the dirty job for a foreign power or rival company. We make several assumptions in the Model, to make it more synthetic:

- We assume one Defender. The Attacker's nodes do not represent a specific attacker, but a generalization of potential criminal organizations that represent business-oriented APTs, guided mostly by monetary incentives.
- We assume an atomic attack (the attacker makes one action), with several consequences, as well as several residual consequences once the risk treatment strategy is selected.
- The Defender and Attacker costs are deterministic nodes.
- We avoid detection-related activities or uncertainties to simplify the Model. Thus, the attack is always detected and the Defender is always able to respond to it.
- The scope of the Model is an assessment activity prior to any attack, as a risk assessment exercise to support incident handling planning.
- The agents are expected utility maximizers.
- The Model is discrete.

By adapting the proposed template in Figure 1, we may generalize most of the above assumptions to the cases at hand.

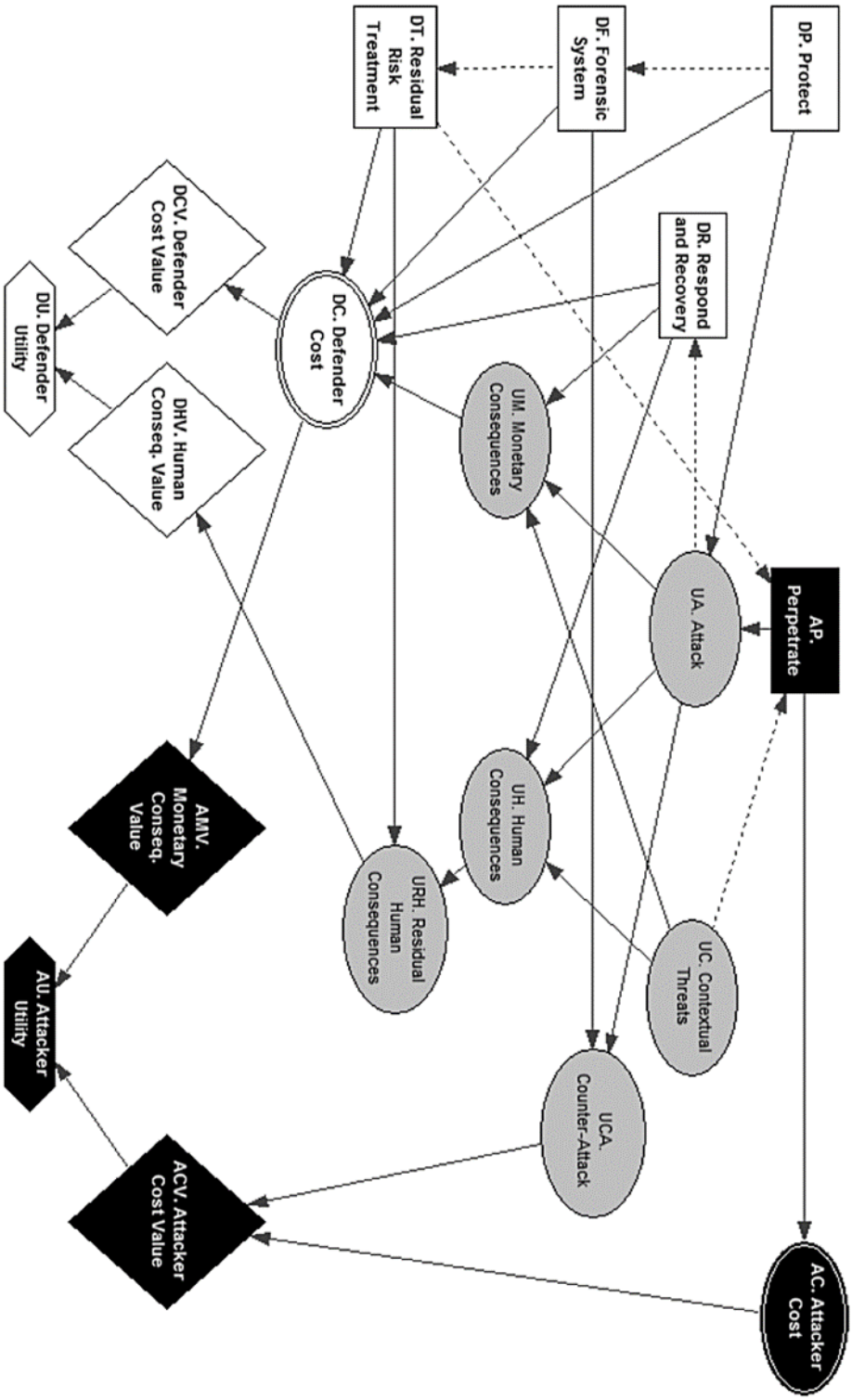

Figure 1. MAID of the ARA Model for O&G drilling cybersecurity.

2.2.1 Defender Decision and Utility Nodes

The Defender nodes, in white, are:

- *Protect* (*DP*) decision node. The Defender selects among security measures portfolios to increase protection against an Attack, e.g., access control, encryption, secure design, firewalls, or personal training and awareness.
- *Forensic System* (*DF*) decision node. The Defender selects among different security measures portfolios that may harm the Attacker, e.g., forensic activities that enable prosecution of the Attacker.
- *Residual Risk Treatment* (*DT*) decision node. This node models Defender actions after the assessment of other decisions made by the Defender and the Attacker. They are based on the main risk treatment strategies excluding risk mitigation, as they are carried on thorugh the Protect and the Respond and Recovery nodes: avoiding, sharing, or accepting risk. This node must be preceded by the Protect defender decision node, and it must precede the Attack uncertainty node (the residual risk assessment is made in advance).
- *Respond and Recovery* (*DR*) decision node. The Defender selects between different response and recovery actions after the materialization of the attack, trying to mitigate the attack consequences. This will depend on the attack uncertainty node.
- *Defender Cost* (*DC*) deterministic node. The costs of the decisions made by the Defender are deterministic, as well as the monetary consequences of the attack (the uncertainty about such consequences is solved in the Monetary Consequences node). In a more sophisticated model, most of the costs could be modeled as uncertain nodes. This node depends on all decision nodes of the Defender and the Monetary Consequences uncertainty node.
- *Value Nodes* (*DCV* and *DHV*). The Defender evaluates the consequences and costs, taking into account her risk attitude. They depend on the particular nodes evaluated at each Value node.
- *Utility Nodes* (*DU*). This node merges the Value nodes of the Defender. It depends on the Defender's Value nodes.

The Decision nodes are adapted to the typical risk management steps, incorporating ways of evaluating managing sound organizational cybersecurity strategy, which takes into account the business implications of security controls, and prepare the evaluation of risk consequences. Related work (Section 1.2) on security costs and investments could incorporate further complexities underlying the above nodes.

2.2.2 Attacker Decision and Utility Nodes

The Attacker nodes, in black, are:

- *Perpetrate* (AP) decision node. The [generic] Attacker decides whether he attacks or not. It could be useful to have a set of options for a same type of attack (e.g., preparing a quick and cheap attack, or a more elaborated one with higher probabilities of success). It should be preceded by the Protect and Residual Risk Treatment decision nodes, and might be preceded by the Contextual Threat node (in case the Attacker observes it).
- *Attacker Cost* (*AC*) deterministic node. Cost of the Attacker decisions. Preceded by the Perpetrate decision node.

- *Value Nodes* (*AMV* and *ACV*). The Attacker evaluates the different consequences and costs, taking into account his risk attitude. They depend on the deterministic or uncertainty nodes evaluated at each Value node.
- *Utility Nodes* (*AU*). It merges the Value nodes of the Attacker to a final set of values. It must depend on the Attacker's Value nodes.

These nodes help in characterizing the Attacker, avoiding the oversimplification of other approaches. Additionally, the Defender has uncertainty about the Attacker probabilities and utilities. This is propagated over their nodes, affecting the Attacker expected utility and optimal alternatives, which are random. Such distribution over optimal alternatives is our forecast for the Attacker's actions.

2.2.3 Uncertainty Nodes

The uncertainty nodes in grey are:

- *Contextual Threats* (*UC*) uncertainty node. Those threats (materialized or not) present during the Attack. The Attacker may carry out a selected opportunistic Attack (e.g. hurricanes or a critical moment during drilling).
- *Attack* (*UA*) uncertainty node. It represents the likelihood of the attack event, given its conditioning nodes. It depends on the Perpetrate decision node, and on the Protect decision node.
- *Consequences* (*UM* and *UV*) uncertainty node. It represents the likelihood of different consequence levels that a successful attack may lead to. They depend on the Attack and Contextual Threat uncertainty nodes, and on the Respond and Recovery decision node.
- *Residual Consequences* (*URH*) uncertainty node. It represents the likelihood of different consequence levels after applying residual risk treatment actions. They depend on the Consequence node modelling the same type of impact (e.g., human, environmental, or reputation).
- *Counter-Attack* (*UCA*) uncertainty node. Possibility, enabled by a forensic system, to counter-attack and cause harm to the Attacker. Most of the impacts may be monetized. It depends on the Forensic System decision node.

Dealing with the uncertainties and complexities and obtaining a probability distribution for these nodes could be hard. Some of the methodologies and findings proposed in the sections 1.1 and 1.2 are tailored to deal with some of these complexities. Using them, the Model proposed in this paper could lead to limit the uncertainties in cybersecurity elements such as vulnerabilities, controls, consequences, attacks, attacker behavior, and risks. This will enable achieving simplification, through the proposed Model, without limiting the understanding of the complexities involved, and a sounder organizational cybersecurity.

3 Example

We present a numerical example of the previous Model tailored to a generic decision problem prototypical of a cybersecurity case that may arise in O&G offshore rig using drilling CS. The model specifies a case in which the driller makes decisions to prevent and respond to a cyber attack perpetrated by a criminal organization with APT capabilities, in the context of offshore drilling and drilling CS. The data employed in this example are just plausible figures helpful to provide an overview of the problems

that drilling cybersecurity faces. Carrying on the assessment that the Model enables may be helpful for feeding a threat knowledge base, incident management procedures or incident detection systems.

The context is that of an offshore drilling rig, a floating platform with equipment to drill a well through the seafloor, trying to achieve a hydrocarbon reservoir. Drilling operations are dangerous and several incidents may happen in the few months (usually between 2 or 4) that the entire operation may last. As OT, drilling CS may face most of the challenges presented in Section 1.1 (including being connected to Enterprise networks, an entry path for attackers) in the context of high-risk incidents that occur in offshore drilling.

3.1 Agent Decisions

3.1.1 Defender Decisions

The Defender has to make three decisions in advance of the potential attack. In the Protect decision node (DP), the Defender must decide whether she invests in additional protection: if the Defender implements additional protective measures, the system will be less vulnerable to attacks. In the Forensic System decision node (DF), the Defender must decide whether she implements a forensic system or not. Implementing it enables the option of identifying the Attacker and pursuing legal or counter-hacking actions against him. The Residual Risk Treatment decision node (DT) represents additional risk treatment strategies that the Defender is able to implement: avoiding (aborting the entire drilling operation to elude the attack), sharing (buying insurance to cover the monetary losses of the attack), and accepting the risk (inheriting all the consequences of the attack, conditional on to the mitigation decisions of DP, FD, and DR).

Additionally, the Respond and Recovery decision node (DR) represents the Defender's decision between continuing and stopping the drilling operations as a reaction to the attack. Continuing the drilling may lead to worsen the consequences of the attack, whereas stopping the drilling will incur in higher costs due to holding operations. This is a major issue for drilling CS. In general, critical equipment should not be stopped, since core operations or even the safety of the equipment or the crew may be compromised.

3.1.2 Attacker Decisions

For simplicity, in the Perpetrate decision node (AP) the Attacker decides whether he perpetrates the attack or not, although further attack options could be added. In this example, the attack aims at manipulating the devices directly under control of physical systems with the purpose of compromising drilling operations or harming equipment, the well, the reservoir, or even people.

3.2 Threat Outcomes and Uncertainty

3.2.1 Outcomes and Uncertainty during the Incident

The Contextual Threats uncertainty node (UC) represents the existence of riskier conditions in the drilling operations (e.g., bad weather or one of the usual incidents during drilling), which can clearly worsen the consequences of the attack. In this scenario, the Attacker is able to know, to some extent, these contextual threats (e.g., a weather forecast, a previous hacking in the drilling CS that permits the attacker to read what is going on in the rig).

The Attack uncertainty node (UA) represents the chances of the Attacker of causing the incident. If the Attacker decides not to execute his action, the no attack event will happen. However, in case of perpetration, the chances of a successful attack will be lower if the Defender invests in protective measures (DC node). An additional uncertainty arises in case of materialization of the attack: the possibility to identify and counter-attack the node, represented by the Counter-Attack uncertainty node (UCA).

If the attack happens, the Defender will have to deal with different consequence scenarios. The Monetary (UM) and Human Consequences (UH) nodes represent the chances of different consequences or impact levels that the Defender may face. The monetary consequences refer to all impacts that can be measured as monetary losses, whereas human consequences represent casualties that may occur during an incident or normal operations. However, the Defender has the option to react to the attack by deciding whether she continues or stops the drilling (DR node). If the Defender decides to stop, there will be lower chances of casualties and lower chances of worst monetary consequences (e.g., loss of assets or compensations for injuries or deaths), but she will have to assume the costs of keeping the rig held (one day in our example) to deal with the cyber threat.

3.2.2 Outcomes and Uncertainty in Risk Management Process

The previous uncertainties appear after the Attacker's decision to attack or not. The Defender faces additional relevant uncertainties. She must make a decision between avoiding, sharing, or accepting the risk (DT node). Such decision will determine the final or residual consequences. The final monetary consequences are modeled through the Defender Cost deterministic node (DC node), whose outcome represents the cost of different Defender decisions (nodes DP, DF, DT, and DR). In case of accepting or sharing the risk, the outcome of the DC node will also inherit the monetary consequences of the attack (UM node). Similarly, the outcome of the Residual Human Consequences uncertainty node (URH) is conditioned by the risk treatment decisions (DC node) and, in case of accepting or sharing the risk, it will inherit the human consequences of the attack (UH node). If the Defender decides to avoid the risk, she will assume the cost of avoiding the entire drilling operations and will cause that the crew face a regular death risk rather than the higher death risk of offshore operations. If the Defender shares the risk, she will assume the same casualties as in UH and a fixed insurance payment, but she will avoid paying high monetary consequences. Finally, in case the Defender accepts the risk, she will inherit the consequences from the UM and UH nodes.

The Attacker Cost deterministic node (AC) provides the costs (non-uncertain by assumption) of the decision made by the Attacker. Since he only has two decisions (perpetrate or not), the node has only two outcomes: cost or not. This node could be eliminated, but we keep it to preserve the business semantics within the graphical model.

3.3 Agent Preferences

The Defender aims at maximizing her expected utility, with the utility function being additive, through the Defender Utility node (DU). The Defender key objective is minimizing casualties, but he also considers minimizing his costs (in this example we assume she is risk-neutral). Each objective has its own weight in the utility function.

The objective of the Attacker is to maximize his expected utility, represented by an additive utility function, through the Attacker Utility node (AU). The Attacker key objective is maximizing the monetary consequences for the Defender. We assume that he is risk-averse towards this monetary impact (he prefers ensuring a lower impact than risking the operations trying to get a higher impact). He also

considers minimizing his costs (i.e., being identified and perpetrating the attack). Each of these objectives has its own weight in the utility function, and its own value function. The Attacker does not care about eventual victims.

3.4 Uncertainty about the Opponent Decisions

The Attacker is able to know to some extent the protective decisions of the defender (DP node), gathering information while he tries to gain access to the drilling CS. While knowing if the Defender avoided the risk (avoiding all the drilling operations) is easy, knowing if the Defender chose between sharing or accepting the risk is difficult. The most important factor, the decision between continue or stop drilling in case of an attack, could be assessed by observing the industry or company practices. The Defender may be able to assess also how frequent similar attacks are, or how attractive the drilling rig is for this kind of attacker. In ARA, and from the Defender perspective, the AP node would be an uncertainty node whose values should be provided by assessing the probabilities of the different attack actions, through analyzing the decision problem from the Attacker perspective and obtaining his random optimal alternative.

3.5 Example Values

An annex provides the probability tables of the different uncertainty nodes employed to simulate the example in Genie (Tables 1 to 7). It also provides the different parameters employed in the utility and value functions (Tables 8 to 10). Additionally, the "risk-averse" values for AMV are obtained with $AMV = \sqrt[3]{\frac{DC}{10^7}}$; the "risk-neutral" values for DCV are obtained with $DCV = 1 - \frac{DC}{10^7}$; and, the values for DHV are 0 in case of victims and 1 in case of no victims.

3.6 Evaluation of Decisions

Based on the solution of the example, we may say that the Attacker should not perpetrate his action in case he believes the Defender will avoid or share the risk. However, the Attacker may be interested in perpetrating his action in case he believes that the Defender is accepting the risk. Additionally, the less preventive measures the Defender implements (DP and DT nodes), the more motivated the Attacker would be (if he thinks the Defender is sharing the risk). The Attacker's expected utility is listed in Table 11 in the Annex. The Defender will choose in this example not to implement additional protection (DP node) without a forensic system (DF node). If the Defender believes that she is going to be attacked, then she would prefer sharing the risk (DT node) and stop drilling after the incident (DR node). In case she believes that there will be no attack, she should accept the risk and continue drilling. The Defender's expected utilities are listed in Table T12 in Annex.

Thus, the Defender optimal decisions create a situation in which the Attacker is more interested in perpetrating the attack. Therefore, to affect the Attacker's behavior, the Defender should provide the image that her organization is concerned with safety, and especially that it is going to share risks. On the other hand, if the Attacker perceives that the Defender pays no attention to safety or that she is going to accept the risk, he will try to carry on his attack. The ARA solution for the Defender is the following:

1. Assess the problem from the point of view of the Attacker. The DT and DR nodes are uncertainty nodes since that Defender decisions are uncertain for the Attacker. The Defender must model such nodes in the way that she thinks the Attacker models such uncertainties. In general, perpetrating an attack is more attractive in case the Attacker strongly believes that the Defender is going to accept the risk or is going to continue drilling.

2. Once forecasted the Attacker's decision, the Defender should choose between sharing and accepting the risk. Accepting the risk in case of no attack is better than sharing the risk, but accepting the risk in case of attack is worse.

Thus, the key factor for optimizing the decision of the Defender are her estimations on the uncertainty nodes that represent the DT and DR nodes for the attacker. Such nodes will determine the Attacker best decision, and this decision the Defender best decision.

4 Conclusions and Further Work

We have presented the real problem and extreme consequences that OT cybersecurity in general, and drilling cybersecurity in particular, are facing. We also explained some of the questions that complicate cybersecurity, especially in OT systems. The proposed graphical model provides a more comprehensive, formal and rigorous risk analysis for cybersecurity. It is also a suitable tool, able of being fed by, or compatible with, other more specific models such as those explained in Section 1.

Multi-Agent Influence Diagrams provide a formal and understandable way of dealing with complex interactive issues. In particular, they have a high value as business tools, since its nodes translate the problem directly into business language: decisions, risks, and value. Typical tools employed in widely used risk standards, such as risk matrices, oversimplify the problem and limit understanding. The proposed ARA-based model provides a business-friendly interpretation of a risk management process without oversimplifying its underlying complexity.

The ARA approach permits us to include some of the findings of game theory applied to cybersecurity, and it also permits to achieve new findings. The model provides an easier way to understand the problem but it is still formal since the causes and consequences in the model are clearly presented, while avoiding common knowledge assumptions in game theory.

Our model presents a richer approach for assessing risk than risk matrices, but it still has the security and risk management language. In addition, it is more interactive and modular, nodes can be split into more specific ones. The proposed model can still seem quite formal to business users. However, data can be characterized using ordinal values (e.g., if we only know that one thing is more likely/valuable than other), using methods taken from traditional risk management, employing expert opinion, or using worst case figures considered realistic. The analysis would be poorer but much more operational.

Using the nodes of the proposed model as building blocks, the model could gain in comprehensiveness through adding more attackers or attacks, more specific decision nodes, more uncertainty nodes, or additional consequence nodes, such as environmental impact or reputation. Other operations with significant business interpretation can be done, such as sensitivity analysis (how much the decision-makers should trust a figure) or strength of the influence analysis (which are the key elements).

Its applicability is not exempt of difficulties and uncertainties, but in the same way than other approaches. Further work is needed to verify and validate the model and its procedures (in a similar way to the validation of other ARA-based models [33]), and to identify the applicability and usability issues that may arise. The model could gain usability through mapping only the relevant information to decision-makers (roughly, decisions and consequences) rather than the entire model.

5 Acknowledgments

This work is supported by FP7 Seconomics project 285223. David Rios Insua is grateful to the support of MINECO Riesgos project and the Riesgos-CM program.

References

[1] S. Amin, G. A. Schwartz & S. S. Sastry (2011): *On the interdependence of reliability and security in networked control systems*. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, IEEE, pp. 4078–4083.

[2] R. Anderson & S Fuloria (2010): *Security economics and critical national infrastructure*. In: Economics of Information Security and Privacy, Springer, pp. 55–66.

[3] A. Atzeni, C. Cameroni, S. Faily, J. Lyle & I Fléchais (2011): *Here's Johnny: A methodology for developing attacker personas*. In: Availability, Reliability and Security (ARES), 2011 Sixth International Conference on, IEEE, pp. 722–727.

[4] M. G Balchanos (2012): *A probabilistic technique for the assessment of complex dynamic system resilience*.

[5] Defense Science Board (2013): *Task Force report: Resilient military systems and the advanced cyber threat*. Department of Defense.

[6] J. F. Brenner (2013): *Eyes wide shut: The growing threat of cyber attacks on industrial control systems*. Bulletin of the atomic scientists (1974) 69(5), pp. 15–20.

[7] E. Byres & J Lowe (2004): *The myths and facts behind cyber security risks for industrial control systems*. In: Proceedings of the VDE Kongress, 116.

[8] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig & S. Sastry (2009): *Challenges for securing cyber physical systems*. In: Workshop on future directions in cyber-physical systems security.

[9] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang & S. Sastry (2011): *Attacks against process control systems: risk assessment, detection, and response*. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, pp. 355–366.

[10] A. A. Cárdenas, S. Amin & S. Sastry (2008): *Research challenges for the security of control systems.* In: HotSec.

[11] Conning (2013): *ADVISE enterprise risk modeler*. Available at `https://www.conning.com/risk-and-capital-management/software/advise.html`. Retrieved: 12/13/2013.

[12] L Cox (2008): *What's wrong with risk matrices?* Risk analysis 28(2), pp. 497–512.

[13] R. Dantu, P. Kolan, R. Akl & K Loper (2007): *Classification of attributes and behavior in risk management using bayesian networks*. In: Intelligence and Security Informatics, 2007 IEEE, IEEE, pp. 71–74.

[14] R. Dantu, P. Kolan & J Cangussu (2009): *Network risk management using attacker profiling*. Security and Communication Networks 2(1), pp. 83–96.

[15] D. Florêncio & C Herley (2013): *Where do all the attacks go?* In: Economics of Information Security and Privacy III, Springer, pp. 13–33.

[16] M. Frigault, L. Wang, A. Singhal & S Jajodia (2008): *Measuring network security using dynamic bayesian network*. In: Proceedings of the 4th ACM workshop on Quality of protection, ACM, pp. 23–30.

[17] A. Giani, S. Sastry, K. H. Johansson & H Sandberg (2009): *The VIKING project: an initiative on resilient control of power networks*. In: Resilient Control Systems, 2009. ISRCS'09. 2nd International Symposium on, IEEE, pp. 31–35.

[18] S. N. Hamilton, W. L. Miller, A. Ott & O. Saydjari (2002): *Challenges in applying game theory to the domain of information warfare*. In: 4th Information survivability workshop (ISW-2001/2002), Vancouver, Canada.

[19] E. M. Hutchins, M. J. Cloppert & R. M. Amin (2011): *Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains*. Leading Issues in Information Warfare & Security Research 1, p. 80.

[20] Gartner IT: *Gartner IT Glossary*. Available at `http://www.gartner.com/it-glossary/operational-technology-ot`. Retrieved: 12/13/2013.

[21] B. Johnson, J. Grossklags, N. Christin & J Chuang (2012): *Nash equilibria for weakest target security games with heterogeneous agents.* In: Game Theory for Networks, Springer Berlin Heidelberg, pp. 444–458.

[22] I. Kotenko & M Stepashkin (2006): *Attack graph based evaluation of network security*. In: *Communications and Multimedia Security*, Springer, pp. 216–227.

[23] Decision Systems Laboratory: *GeNIe*. Available at `http://genie.sis.pitt.edu/`.

[24] Z. Li, Q. Liao & A Striegel (2009): *Botnet economics: uncertainty matters*. In: *Managing Information Risk and the Economics of Security*, Springer, pp. 245–267.

[25] P. Liu, W. Zang & M Yu (2005): *Incentive-based modeling and inference of attacker intent, objectives, and strategies*. *ACM Transactions on Information and System Security (TISSEC)* 8(1), pp. 78–118.

[26] Command Five Pty Ltd (2011): *Advanced persistent threats: A decade in review*. Available at `http://www.commandfive.com/papers/C5_APT_ADecadeInReview.pdf`. Retrieved: 12/13/2013.

[27] M. S. Lund, B. Solhaug & K Stolen (2011): *Model-driven risk analysis: the CORAS approach*. Springer.

[28] S. Mauw & M Oostdijk (2006): *Foundations of attack trees*. In: *Information Security and Cryptology-ICISC 2005*, Springer, pp. 186–198.

[29] J. Merrick & G. S. Parnell (2011): *A comparative analysis of PRA and intelligent adversary methods for counterterrorism risk management*. *Risk Analysis* 31(9), pp. 1488–1510.

[30] C. Muehrcke, E. V. Ruitenbeek, K. Keefe & W. H. Sanders (2010): *Characterizing the behavior of cyber adversaries: The means, motive, and opportunity of cyberattacks*. IEEE/IFIP International Conference on Dependable Systems and Networks.

[31] D. K. Mulligan & F. B. Schneider (2011): *Doctrine for cybersecurity*. *Daedalus* 140(4), pp. 70–92.

[32] S. L. Pfleeger & R Rue (2008): *Cybersecurity economic issues: Clearing the path to good practice*. *Software, IEEE* 25(1), pp. 35–42.

[33] D. Rios Insua & J. Cano (2013): *Basic models for security risk analysis. SECONOMICS D5.1*. Technical Report.

[34] David Rios Insua, J. Rios & D Banks (2009): *Adversarial risk analysis*. *Journal of the American Statistical Association* 104(486), pp. 841–854.

[35] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya & Q Wu (2010): *A survey of game theory as applied to network security*. In: *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, IEEE, pp. 1–10.

[36] K Sallhammar (2007): *Stochastic models for combined security and dependability evaluation*. Ph.D. thesis, Norwegian University of Science and Technology.

[37] J. C. Sevillano, D Rios Insua & J Rios (2012): *Adversarial risk analysis: The Somali pirates case*. *Decision Analysis* 9(2), pp. 86–95.

[38] Z. Shauk (2013): *Hackers hit energy companies more than others*. Available at `http://fuelfix.com/blog/2013/03/25/electronic-attacks-hit-two-thirds-of-energy-companies-in-study/`. Retrieved: 12/13/2013.

[39] C.-W. Ten, C.-C. Liu & G Manimaran (2008): *Vulnerability assessment of cybersecurity for SCADA systems*. *Power Systems, IEEE Transactions on* 23(4), pp. 1836–1846.

[40] R. C. Thomas, M. Antkiewicz, P. Florer, S. Widup & M Woodyard (2013): *How bad is it?–A branching activity model to estimate the impact of information security breaches*.

[41] P. Xie, J. H. Li, X. Ou, P. Liu & R Levy (2010): *Using Bayesian networks for cyber security analysis*. In: *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, IEEE, pp. 211–220.

[42] B. Zhu, A. Joseph & S Sastry (2011): *A taxonomy of cyber attacks on SCADA systems*. In: *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, IEEE, pp. 380–388.

Table T1. Probability table for UC node.

Riskier conditions	30%
Normal conditions	70%

Table T2. Probability table for UA node.

Attacker's Perpetrate decision	*Perpetrate*		*No perpetrate*	
Defender's Protect decision	*Additional protection*	*Non additional protection*	*Additional protection*	*Non additional protection*
Attack event	5%	40%	0%	0%
No attack event	95%	60%	100%	100%

Table T3. Probability table for UM node.

Attack event	*Attack*				*No attack*			
Contextual Threat event	*Riskier conditions*		*Normal conditions*		*Riskier conditions*		*Normal conditions*	
Defender's Respond and Recovery decision	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*
Lossing 0 $ event	3%	0%	10%	0%	92%	0%	96%	0%
Lossing 0 - 1 Million $ event	12%	85%	20%	90%	7%	97%	4%	99%
Lossing 1 - 5 Million $ event	85%	15%	70%	10%	1%	3%	0%	1%

Table T4. Probability table for UH node.

Attack event	*Attack*				*No attack*			
Contextual Threat event	*Riskier conditions*		*Normal conditions*		*Riskier conditions*		*Normal conditions*	
Defender's Respond and Recovery decision	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*	*Continue drilling*	*Stop drilling*
Non casualties event	96%	99.2%	99.4%	99.96%	99.6%	99.96%	99.9%	99.99%
Casualties event	4%	0.8%	0.6%	0.04%	0.4%	0.04%	0.1%	0.01%

Table T5. Probability table for URH node.

Human Consequences event	*No casualties*			*Casualties*		
Defender's Residual Risk Treatment decision	*Avoid*	*Share*	*Accept*	*Avoid*	*Share*	*Accept*
No casualties event	99.95%	100%	100%	0%	0%	0%
casualties event	0.05%	0%	0%	100%	100%	100%

Table T6. Probability table for UCA node.

Attack event	Attack		No attack	
Defender's Forensic System decision	*Forensic*	*No forensic*	*Forensic*	*No forensic*
No identification event	30%	90%	100%	100%
Identification event	70%	10%	0%	0%

Table T7. Probability table for DC node:

Avoiding the risk	10,000,000 $			
Sharing the risk	500,000 $			
Accepting the risk	*Monetary Consequences event*	0 $	0 - 1,000,000 $	1,000,000 - 5,000,000 $
	Value assigned	0 $	500,000$	2,500,000 $
Additional protection	20,000 $			
Forensic system	10,000 $			
Stop drilling	300,000 $			

Table T8. Weight table for DU node.

Importance of the Costs	5%
Importance of the Human Consequences	95%

Table T9. Value table for ACV node:

Attacker Cost event	*Cost*		*No cost*	
Counter Attack Consequences event	*No identification*	*Identification*	*No identification*	*Identification*
Value	0.75	0	1	0.25

Table T10. Weight table for AU node.

Importance of the costs	3%
Importance of the Monetary Consequences on the Defender	97%

Table T11. Attacker expected utilities (in black the highest among the different Attacker's decisions).

DP node	*DF node*	*DT node*	*UC node*	*Defender continues drilling*		*Defender stops drilling*	
				Perpetrate decision	*Non perpetrate decision*	*Perpetrate decision*	*Non perpetrate decision*
Additional protection	*Forensic*	*Avoid*	*Riskier conditions*	**1**			
			Normal conditions	**1**			
		Share	*Riskier conditions*	0.56074	**0.56903**	0.61138	**0.61966**
			Normal conditions	0.56074	**0.56903**	0.61138	**0.61966**
		Accept	*Riskier conditions*	0.36484	0.35433	0.61728	0.62458
			Normal conditions	0.35170	0.34293	0.61375	0.62130
	No forensic	*Avoid*	*Riskier conditions*	**1**			
			Normal conditions	**1**			
		Share	*Riskier conditions*	0.55938	**0.56699**	0.61060	**0.61821**
			Normal conditions	0.55938	**0.56699**	0.61060	**0.61821**
		Accept	*Riskier conditions*	0.34461	0.33241	0.61653	0.62315
			Normal conditions	0.33055	0.32013	0.61299	0.61986
No additional protection	*Forensic*	*Avoid*	*Riskier conditions*	1			
			Normal conditions	1			
		Share	*Riskier conditions*	0.55116	**0.56496**	0.60295	**0.61675**
			Normal conditions	0.55116	**0.56496**	0.60295	**0.61675**
		Accept	*Riskier conditions*	0.45634	0.29898	0.61588	0.62173
			Normal conditions	0.42794	0.28532	0.61058	0.61841
	No Forensic	*Avoid*	*Riskier conditions*	1			
			Normal conditions	1			
		Share	*Riskier conditions*	0.55442	**0.56282**	0.60690	**0.61530**
			Normal conditions	0.55442	**0.56282**	0.60690	**0.61530**
		Accept	*Riskier conditions*	0.32392	0.07465	0.61990	0.62030
			Normal conditions	0.28286	0.05131	0.61456	0.61696

Table T12. Defender expected utilitoes (in black the highest among the different Defender's decisions).

DP node	*DF node*	*DT node*	*DR node*	*Possible events*			
				Riskier conditions		*Normal conditions*	
				Attack event	*Non attack event*	*Attack event*	*Non attack event*
Additional protection	*Forensic*	*Avoid*	*Continue drilling*	0.91154	0.94573	0.94383	0.94858
			Stop drilling	0.94193	0.94915	0.94915	0.94943
		Share	*Continue drilling*	0.95935	0.99355	0.99165	0.99640
			Stop drilling	0.98825	0.99547	0.99547	0.99576
		Accept	*Continue drilling*	0.95092	0.99575	0.98490	0.99880
			Stop drilling	0.98675	0.99517	0.99447	0.99566
	No forensic	*Avoid*	*Continue drilling*	0.91154	0.94573	0.94383	0.94858
			Stop drilling	0.94193	0.94915	0.94915	0.94943
		Share	*Continue drilling*	0.95940	0.99360	0.99170	0.99645
			Stop drilling	0.98830	0.99552	0.99552	0.99581
		Accept	*Continue drilling*	0.95097	0.99580	0.98495	0.99885
			Stop drilling	0.98680	0.99522	0.99452	0.99571
No additional protection	*Forensic*	*Avoid*	*Continue drilling*	0.91154	0.94573	0.94383	0.94858
			Stop drilling	0.94193	0.94915	0.94915	0.94943
		Share	*Continue drilling*	0.95945	0.99365	0.99175	0.99650
			Stop drilling	0.98835	0.99557	0.99557	0.99586
		Accept	*Continue drilling*	0.95102	0.99585	0.98500	0.99890
			Stop drilling	0.98685	0.99527	0.99457	0.99576
	No Forensic	*Avoid*	*Continue drilling*	0.91154	0.94573	0.94383	0.94858
			Stop drilling	0.94193	0.94915	0.94915	0.94943
		Share	*Continue drilling*	0.95950	0.99370	0.99180	0.99655
			Stop drilling	**0.98840**	0.99562	**0.99562**	0.99591
		Accept	*Continue drilling*	0.95107	**0.99590**	0.98505	**0.99895**
			Stop drilling	0.98690	0.99532	0.99462	0.99581