Security (hyper-)properties in workflow systems: From specification to verification*

Thomas Bauereiss
Advisor: Dieter Hutter

German Research Center for Artificial Intelligence (DFKI)
Bremen, Germany

Abstract. Provable security guarantees for software systems are highly desirable. Our work aims at improving and integrating existing formal verification techniques into a framework for the specification and verification of typical security requirements of large-scale, distributed workflow systems. Challenges include the uniform modelling of different types of security requirements, the decomposition of global security requirements into requirements on subcomponents, and the refinement of an abstract specification towards an implementation. We focus our attention on workflow management systems due to their interesting security requirements and the widespread use of model-driven techniques in this area (e.g. using BPMN diagrams). We build upon existing verification techniques for a specific notion of information flow security, and intend to apply our results to concrete example systems such as a secure web-based conference management system.

1 Motivation

As computer systems grow increasingly complex and pervade more and more aspects of everyday life, reliable security properties become increasingly important. In large, distributed systems that facilitate the collaboration of multiple users there are different types of security requirements that need to be satisfied by the subsystems. The confidentiality and integrity of data items that are processed in the system needs to be protected, and there are security requirements regarding the users involved in the process, e.g. separation of duty constraints, requiring that at least two users must agree on a joint decision before the corresponding action can be taken.

Addressing these security requirements already in early phases of the development process avoids costly changes to the architecture and design of the system in later phases. Tool support for (semi-)automatic analysis of security aspects is needed to cope with the increasing complexity of computer systems. This analysis should be based on well-founded models and theories of computer security in order to allow reliable guarantees of security 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."

Existing approaches to workflow security typically map security requirements to access control configurations (e.g. [5, 22]), while some formalise security requirements as LTL formulas and employ model-checking for verification (e.g. [2, 19]). This is suitable for safety or liveness properties, but it is insufficient for many notions of information flow security that can be seen as hyperproperties [7]. Information flow control goes beyond mere access control by taking into account the complete behaviour of the system, thereby preventing not only direct but also implicit information leaks via observation of the system. Formally, while safety and liveness properties correspond to sets of traces (i.e. system runs), hyperproperties have been defined in [7] as sets of sets of traces, or equivalently, as predicates on complete systems. In particular, possibilistic information flow predicates can be seen as closure predicates on trace sets [14], e.g. requiring that for each trace with a confidential event, an alternative trace without the event must exist that yields the same visible observations.

Enforcing a safety property by removing traces that violate the property can potentially destroy information flow security: For example, consider a secure workflow system where an additional separation of duty constraint between a confidential and a non-confidential activity is to be enforced. Someone who can observe the non-confidential activity and sees a certain user perform it can deduce that this user has not participated in the confidential activity. This might be an information leak in itself (if anonymity is a concern), and if different users are allowed to perform different actions it might even leak information about the exact actions that could have been performed in the confidential activity. Hence, there can be subtle interrelations between different types of security requirements, and our aim is to develop a framework where they can be treated in an integrated way and refined from the specification to the implementation level.

2 Aims and Objectives

We focus on workflow management systems due to their interesting security requirements. In [4], we used a hiring workflow as a running example, where medical data about applicants has to be kept confidential and separation of duty constraints between different medical officers have to be enforced. A developer who wants to implement such a system needs to map the security requirements to a secure implementation in some way. Our vision is a step-wise development process, starting from high-level specifications of the system (e.g. a BPMN workflow diagram) and its security requirements (e.g. as annotations in the workflow diagram) that are mapped to a formal model. Refinement techniques and tools then support the developer in performing refinement steps towards an implementation in such a way that security properties established on the abstract level are preserved by the refinement. In such a refinement step, the developer can replace an activity in a workflow by a subprocess, or refine behavioural specifications of atomic activities. We want this development process to be well-founded on formal models and theories of security so that the resulting implemented system

has provable security properties. Our goal is to build upon existing formal verification techniques and identify and close existing gaps along the way from an initial workflow specification to a verified implementation. In particular, we aim at the following contributions:

- A framework for specifying workflow systems and their relevant security requirements: Workflow security is typically understood in terms of access control, with some exceptions [23, 1]. However, [23] does not state the actual information flow property that it checks in a declarative, mechanism-independent way, while [1] focuses on a specific notion of information flow that can be checked by a structural analysis of a Petri net representation of the workflow. We choose to build upon the MAKS framework for possibilistic information flow [14], as it unifies several existing notions of information flow from the literature. We model workflows as state-event systems suitable for verification in the MAKS framework, and map data requirements to information flow predicates and process requirements to safety properties.
- A verification framework for both data and process requirements: We adapt an existing methodology for compositional verification of information flow security, and we propose to use a compositional approach to verify the compatibility of information flow predicates and safety properties [3]. We found that this leads to intuitive and reusable results in the cases that we have considered, and we believe it can be a useful complement to existing approaches to security-preserving refinement, e.g. [15].
- An improved technique for action refinement in MAKS [10] in order to move from a specification closer to the implementation level: [10] allows to replace atomic abstract events with sequences of more concrete events, and gives sufficient conditions for the preservation of security by such a refinement. However, it requires configuration structure semantics in order to deal with concurrency, and the conditions for preservation of security are rather strict. We see room for improvement wrt. relaxing these conditions, possibly integrating insights from related approaches for other formalisms [6, 17, 21].

On a more technical level, we develop the above techniques not only using pen and paper, but also within the interactive theorem prover Isabelle/HOL based on an existing formalisation of the MAKS framework. This serves to verify our results and also to connect to other formalisations and tools available for Isabelle/HOL, e.g. the large HOL library for the specification of software in terms of functional programs, formalisations of operational semantics of some programming languages, e.g. [12], tool support for data refinement [13], or code generation from specifications to languages such as Scala [9]. Ideally, our formalised theories will become the foundation of an analysis tool for workflow security that can be integrated into a workflow modelling tool.

3 Research plan

Our research is on formally specifying and verifying security properties of software systems, hence the research methodology centers around the formalisation

of the system and desired properties, and the development of proof techniques for verification. Our work done so far has focused on modelling and verifying security at the abstract level, so future work will focus on the aspects of refinement, tool support, and evaluation.

3.1 Work done to date

System model: In [4], we have formalised a workflow management system on an abstract level in terms of a composition of subsystems representing the activities in the workflow. This facilitates the distributed deployment of workflow systems, e.g. using web services, and for compositional verification of security. Activities in the workflow are mapped to state-event systems that pass on data items and control flow triggers by sending messages to each other. The design decisions regarding the interactions of these activities were inspired by the BPMN standard, which describes the execution semantics of the control flow, for example, in terms of tokens that are passed from one activity to the next one, corresponding to trigger messages in our model. Essentially, our model captures a basic subset of BPMN, and it should be straightforward to add more complex aspects such as exception handling.

Security properties: In [4], we have mapped data requirements to information flow predicates. System events representing the input of a confidential data item are classified as confidential events, and the events belonging to activities with a lower security classification as visible. The information flow predicates then formalise the requirement that someone who observes or participates in visible activities cannot deduce information about the occurence or non-occurence of confidential events and, hence, the values of confidential data items.

We model process requirements such as separation of duty as safety properties, characterising the sets of execution traces that do not violate the security requirement. This allows us to formally model such security requirements on the abstract level without having to refer to implementation details of the enforcement mechanism.

Compositional verification: For verification of information flow security, we apply a decomposition methodology [11] to split up the overall security requirement into requirements for the individual components and then prove that they satisfy these requirements using an unwinding technique.

We propose to use compositionality also for the integrated verification of information flow security and safety properties [3]. A safety property can be enforced using an execution monitor that runs in parallel with the target system and inhibits or modifies executions that would violate the safety property [20]. We can analyse such a monitor and verify that it does not leak confidential information under certain conditions, and then compose it with the target system. The composed system satisfies the safety property, and the compositionality theorems of the MAKS framework give us sufficient conditions under which this composition preserves information flow security. We demonstrate this approach for separation of duty and for ordered delivery of messages in [3].

3.2 Future work

Refinement: This work so far only takes into account one single level of abstraction. As discussed above, action refinement can be used to map from one level of abstraction to a more concrete one while retaining the security properties established on the abstract level. Eventually, we want to reach the implementation level using such a refinement. This means relating abstract traces to concrete program executions as well as abstract values to concrete data structures, e.g. XML documents. Relating an abstract specification and an implementation in a concrete programming language requires a formal semantics of the language in terms of execution traces, and, preferably, a security type system for establishing the security of programs. There are semantics for realistic languages formalised in Isabelle, e.g. a subset of Java [12]. For a simple while-language, a relation between MAKS security predicates and a language-based notion of information flow security that can be checked using a security type system has been established in [16].

An alternative approach to is to generate an executable implementation from a sufficiently detailed specification using a (trusted) code generator. A potential target language is Scala, as Isabelle supports code generation [9] for it, and the Akka library available for Scala could be used to implement the activities in a workflow system as actors in an actor system.

We will investigate these approaches, formalise a suitable refinement technique in Isabelle, and integrate it into our workflow formalisation.

Evaluation: We aim to evaluate our results in example scenarios. Within the scope of the DFG priority programme "Reliably Secure Software Systems" we currently collaborate with other research projects on a joint reference scenario on Web-based workflow management systems. The concrete example application is a Web-based conference management system, with security requirements such as confidentiality of submissions, anonymity of reviewers, and separation of duty between reviewers and authors. We intend to evaluate our techniques in this scenario and hope to benefit from the collaboration with the other projects, working on aspects such as model-checking information flow security [8] or language-based noninterference [18].

Another demonstrator could be the integration of our techniques into an open-source business process modeling tool, so that a developer can model a workflow, annotate it with security requirements, add or refine behavioural specifications or code to activities in the workflow, and generate an executable implementation. On the basis of our formal semantics for workflows and the verification techniques we employ, such a tool could alert developers to security problems already during specification and throughout the refinement process, guiding them to a provably secure workflow system.

In the long term, we hope that our work on the formal foundations of workflow security and tools building upon them will contribute to increased scalability and adoption of formal methods for the engineering of workflow systems with strong, provable security guarantees.

References

1. Accorsi, R., Lehmann, A.: Automatic information flow analysis of business process models. In: BPM. LNCS, vol. 7481, pp. 172–187. Springer (2012)
2. Arsac, W., Compagna, L., Pellegrino, G., Ponta, S.E.: Security validation of business processes via model-checking. In: ESSoS. No. 6542 in LNCS, Springer (2011)
3. Bauereiss, T., Hutter, D.: Compatibility of safety properties and possibilistic information flow security in MAKS. In: Proc. IFIP SEC 2014, to appear (2014)
4. Bauereiss, T., Hutter, D.: Possibilistic information flow security of workflow management systems. In: GraMSec'14, to appear in EPTCS (2014)
5. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur. 2(1), 65–104 (Feb 1999)
6. Bossi, A., Piazza, C., Rossi, S.: Action refinement in process algebra and security issues. In: LOPSTR 2007, pp. 201–217. No. 4915 in LNCS, Springer (2008)
7. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security 18(6), 1157–1210 (2010)
8. Dimitrova, R., Finkbeiner, B., Kovács, M., Rabe, M., Seidl, H.: Model checking information flow in reactive systems. In: VMCAI (2012)
9. Haftmann, F., Nipkow, T.: A code generator framework for Isabelle/HOL. In: Theorem Proving in Higher Order Logics: Emerging Trends (2007)
10. Hutter, D.: Possibilistic information flow control in MAKS and action refinement. In: ETRICS, pp. 268–281. No. 3995 in LNCS, Springer (2006)
11. Hutter, D., Mantel, H., Schaefer, I., Schairer, A.: Security of multi-agent systems: A case study on comparison shopping. J. Applied Logic 5(2), 303–332 (Jun 2007)
12. Klein, G., Nipkow, T.: A machine-checked model for a java-like language, virtual machine, and compiler. ACM Trans. Program. Lang. Syst. 28(4), 619–695 (2006)
13. Lammich, P.: Automatic data refinement. In: Interactive Theorem Proving, pp. 84–99. No. 7998 in LNCS, Springer (Jan 2013)
14. Mantel, H.: Possibilistic definitions of security-an assembly kit. In: CSFW (2000)
15. Mantel, H.: Preserving information flow properties under refinement. In: IEEE Security & Privacy. pp. 78–91. IEEE (2001)
16. Mantel, H., Sabelfeld, A.: A unifying approach to the security of distributed and multi-threaded programs. Journal of Computer Security 11(4), 615–676 (2003)
17. Martinelli, F., Matteucci, I.: Preserving security properties under refinement. In: SESS. pp. 15—21. ACM (2011)
18. Popescu, A., Hölzl, J., Nipkow, T.: Formal verification of language-based concurrent noninterference. Journal of Formalized Reasoning 6(1), 1–30 (2013)
19. Schaad, A., Lotz, V., Sohr, K.: A model-checking approach to analysing organisational controls in a loan origination process. In: SACMAT'06. ACM (2006)
20. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. 3(1), 30–50 (Feb 2000)
21. Seehusen, F., Stølen, K.: Maintaining information flow security under refinement and transformation. In: FAST 2007. LNCS 4691, pp. 143–157. Springer (2007)
22. Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. J. Syst. Architect. 55(4), 211–223 (2009)
23. Yang, P., Lu, S., Gofman, M.I., Yang, Z.: Information flow analysis of scientific workflows. Journal of Computer and System Sciences 76(6), 390–402 (Sep 2010)