Identifying Confidentiality Violations in Architectural Design Using Palladio

Stephan Seifermann[a], Maximilian Walter[a], Sebastian Hahner[a], Robert Heinrich[a] and Ralf Reussner[a]

[a]KASTEL – Institute of Information Security and Dependability, Karlsruhe Institute of Technology (KIT), Germany

Abstract

Meeting confidentiality requirements in software systems is vital for organizations. Considering confidentiality in early development phases such as the architectural design phase is beneficial compared to late phases such as the implementation because fixing design issues is more cost-efficient in early phases. This tutorial introduces an approach for modeling and statically analyzing confidentiality in software architectures within the Palladio tool suite. Besides foundational knowledge, the tutorial provides a practical hands-on session using the tool. The goal is to show that it is already possible to consider confidentiality in the early design process and that this consideration can be integrated into existing architectural design tools.

Keywords

Confidentiality, Architectural Design, Palladio

1. Motivation

The importance of security in software systems continuously increases together with the connectedness of systems and legal obligations. Breaches of confidentiality, which is an aspect of security, have a significant impact on business because of a loss of business value [1] or high fines [2, 3]. Additionally, many users are willing to change a service provider to increase the confidentiality of their data [4]. Therefore, protecting data confidentiality is vital for organizations.

It is necessary to consider confidentiality in all development phases and as early as possible, because many violations trace back to the software design [5, 6], which also includes early designs such as software architectures. To avoid high effort for later fixes, it is necessary to already identify and address these issues in the software architecture [7]. Even if this line of arguments is clear or even obvious to many, there is still a low adoption of security tools during the software design or architecture [8, 9, 10, 11]. There are various reasons to this low adoption but missing awareness and integration into existing workflows are two of them.

The goal of the tutorial is to raise awareness and to make clear that approaches, which blend into existing tools and processes, exist. We would like to show this by presenting our confidentiality modeling and analysis approach integrated into the Palladio tool suite [12]. The Palladio integration extends the modeling language by means for expressing data processing. A automated transformation translates the Palladio model into an analysis model that determines rel-

Växjö'21: European Conference on Software Architecture, September 13–17, 2021, Växjö, Sweden

stephan.seifermann@kit.edu (S. Seifermann);
maximilian.walter@kit.edu (M. Walter);
sebastian.hahner@kit.edu (S. Hahner);
robert.heinrich@kit.edu (R. Heinrich);
ralf.reussner@kit.edu (R. Reussner)

CEUR Workshop Proceedings (CEUR-WS.org)

evant data properties by simulating the effect of data processing on data. Eventually, data properties are compared with expected properties and the identified problems are reported back to the user.

2. Tutorial Overview

In this tutorial, we show how static analyses of software architectures can reveal violations of access control policies. The threats considered in the analyses are that the software architecture itself violates such policies by its structure, behavior, deployment or intended usage.

The tutorial is built around our modeling and analysis approach [13, 14]. The approach is integrated into the Palladio tool suite [12] and makes use of recently introduced data-oriented interfaces [15]. The approach has already been used in various contexts, so we consider it mature enough for the presentation as part of the tutorial. For instance, the approach has been used to identify read and written data as part of an alignment process between business processes and software architectures focused on access rights [16, 17] as well as for deriving properties of processed data that can speed up runtime analyses [18].

The tutorial consists of three parts: First, we introduce the modeling concepts to represent system behavior that can be analyzed for access control later. This includes a hands-on session on modeling using our Palladio tooling. Second, we explain how to formulate an access control analysis using a previously published domain-specific language [19] and how to interpret the results. Again, this includes a hands-on session on analyzing the previously modeled system using our Palladio tooling. We recap all required foundational knowledge, so no expertise on security or Palladio is required. Third, we will briefly recap the contents of the tutorial, give an outlook on ideas and first results of future work.

3. Tutorial Material

The tutorial will use slides, ready to use Palladio tooling and example models. All material will be published on the companion website of the tutorial[1] and in a data set [20].

The example discussed in the tutorial is the TravelPlanner system known from the iFlow approach [21]. The interactions in the system are visualized by the sequence diagram in Figure 1. Simply said, a user looks for a flight and books the flight by additionally passing credit card information (ccd) to the system. The system consists of the smartphone apps *TravelPlanner* and *CreditCardCenter*. The travel planner mediates between the user and a travel agency, as well as an airline. The credit card center safely stores the credit card information of the user. The *TravelAgency* supports the search for flights and receives a comission from the airline after a booking. The *Airline* provides information about flights and supports booking flights. In the corresponding publication [21], the confidentiality requirement is given by an information flow policy. Transmitted data has a classification level and participants have a clearance level. The levels are {*User, Airline, TravelAgency*}, {*User, Airline*} and {*User*} in ascending order. The requirement is that no data must receive at a participant, which has a clearance level lower than the classification level of the data. In the example, the critical part is the transmission of the credit card information, which is classified as *User*, to the airline, which only has a clearance {*User,Airline*}.

In the tutorial, we use a modified version of the scenario described before, which we published as part of a previous publication

[1] https://fluidtrust.github.io/tutorial-ecsa2021/

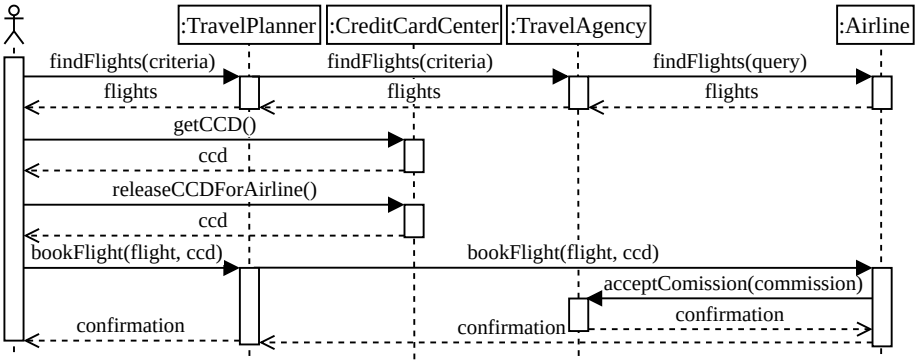

Figure 1: UML sequence diagram illustrating the TravelPlanner system.

[13]. Instead of an information flow policy, we formulate the confidentiality requirements in terms of Role-based Access Control (RBAC). Participants have roles instead of clearance levels and data have associated access rights in terms of roles. The roles are *User*, *TravelAgency* and *Airline*. The requirement is that the intersection between the access rights of data and the roles of a participant must not be empty. Again, the critical part is that credit card data is only accessible to the user but it is sent to the airline during the booking.

In both examples, it is possible to explicitly release the credit card information on behalf of the user. This action reduces the classification level or add the *Airline* role, respectively. The analysis to be defined during the tutorial, detects the violation of the access control policy if the credit card information is passed to the airline without previous release. A missing call to the release service implies an error in the planned behavior of the architecture and has to be fixed.

4. Future Work

We already discovered several interesting areas for further research: Architecture-level confidentiality analyses are affected by uncertainty due to abstraction and lack of information at design time. Therefore, we plan to make uncertainty explicit while modeling, refining and analyzing access control policies [22]. A considerable source of uncertainty is implied by the execution context of the system under analysis. Therefore, it is reasonable to identify and consider relevant aspects of the context [23]. Additional, malicious users or attackers could exist. These could exploit vulnerabilities or policies and therefore should be considered in the future.

Acknowledgements

This work is funded by the DFG (German Research Foundation) – project number 432576552, HE8596/1-1 (FluidTrust) and also supported by funding of the Helmholtz Association (HGF) through the Competence Center for Applied Security Technology (KASTEL) (46.23).

References

[1] H. Weisbaum, Trust in Facebook has dropped by 66 percent since the

Cambridge Analytica scandal, 2018. URL: https://www.nbcnews.com/business/consumer/trust-facebook-has-dropped-51-percent-cambridge-analytica-scandal-n867011, accessed 2021-07-09.

[2] E. Denham, COM0783542, Penatly Notice, UK Information Commissioner's Office, 2020.

[3] E. Denham, COM0804337, Penatly Notice, UK Information Commissioner's Office, 2020.

[4] Cisco Systems, Inc., Consumer Privacy Survey, Technical Report, Cisco, 2019. URL: https://www.cisco.com/c/dam/global/en_uk/products/collateral/security/cybersecurity-series-2019-cps.pdf, accessed 2021-07-09.

[5] R. Kuhn, et al., It Doesn't Have to Be Like This: Cybersecurity Vulnerability Trends, IT Professional 19 (2017) 66–70.

[6] G. McGraw, Software Security - Building Security In, Addison-Wesley Professional, 2006.

[7] F. Shull, et al., What we have learned about fighting defects, in: METRICS, IEEE, 2002, pp. 249–258.

[8] H. Assal, et al., Security in the software development lifecycle, in: SOUPS'18, USENIX Association, 2018, pp. 281–296.

[9] H. Assal, et al., 'Think secure from the beginning': A Survey with Software Developers, in: CHI'19, 2019, pp. 1–13.

[10] J. A. Davis, et al., Study on the Barriers to the Industrial Adoption of Formal Methods, in: Formal Methods for Industrial Critical Systems, LNCS, Springer, 2013, pp. 63–77.

[11] H. Garavel, et al., The 2020 Expert Survey on Formal Methods, in: Formal Methods for Industrial Critical Systems, LNCS, Springer, 2020, pp. 3–69.

[12] R. H. Reussner, et al., Modeling and Simulating Software Architectures - The Palladio Approach, MIT Press, 2016.

[13] S. Seifermann, et al., Data-Driven Software Architecture for Analyzing Confidentiality, in: ICSA'19, IEEE, 2019, pp. 1–10.

[14] S. Seifermann, et al., A Unified Model to Detect Information Flow and Access Control Violations in Software Architectures, in: SECRYPT'21, SCITEPRESS, 2021, pp. 26–37.

[15] D. Werle, et al., Data Stream Operations as First-Class Entities in Component-Based Performance Models, in: ECSA'20, LNCS, Springer, 2020, pp. 148–164.

[16] R. Pilipchuk, et al., Aligning Business Process Access Control Policies with Enterprise Architecture, in: CECC'18, 2018, pp. 17:1–17:4.

[17] R. Pilipchuk, et al., Challenges in Aligning Enterprise Application Architectures to Business Process Access Control Requirements in Evolutional Changes, in: ICE-B'21, ScitePress, 2021, pp. 13–24.

[18] R. Al-Ali, et al., Dynamic Security Rules for Legacy Systems, in: ECSA'19 - Volume 2, 2019, pp. 277–284.

[19] S. Hahner, et al., Modeling data flow constraints for design-time confidentiality analyses, in: ICSA'21 - Companion, 2021, pp. 15–21.

[20] S. Seifermann, et al., Auxiliar material, 2021. doi:10.5281/zenodo.5086778.

[21] K. Katkalov, et al., Model-Driven Development of Information Flow-Secure Systems with IFlow, in: SocialCom'13, 2013, pp. 51–56.

[22] S. Hahner, Architectural access control policy refinement and verification under uncertainty, in: ECSA'21, 2021. Accepted, to appear.

[23] N. Boltz, et al., Context-Based Confidentiality Analysis for Industrial IoT, in: SEAA'20, 2020, pp. 589–596.