Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings (Summary)

Sven Peldszus[1], Katja Tuma[2], Daniel Strüber[2], Jan Jürjens[1,3], Riccardo Scandariato[2]

Abstract: We present our paper published at the 2019 edition of the International Conference on Model Driven Engineering Languages and Systems (MODELS) [Pe19]. During the development of security-critical software, the system implementation must capture the security properties postulated by the architectural design. To iteratively guide the developer in discovering such compliance violations we introduce *automated mappings*. These mappings are created by searching for correspondences between a design-level model (Security Data Flow Diagram) and an implementation-level model (Program Model). We limit the search space by considering name similarities between model elements and code elements as well as by the use of heuristic rules for matching data-flow structures. The automated mappings support the designer in an early discovery of implementation absence, convergence, and divergence with respect to the planned software design as well as the discovery of secure data-flow compliance violations. We provide a publicly available implementation of the approach and its evaluation on five open source Java projects.

1 Introduction

Security threats to software systems are a growing concern in many organizations. Therefore, one needs to consider security early in the design phase, when little is known about the system. Before any new functionality is released, it must be checked that every security assumption made in any of the phases is met. The state of the art for this check in practice are manual code reviews by security experts. Since such reviews are expensive and error-prone, they are only performed on selected code parts, leaving a large leeway for security threats.

In the context of software architecture design, threat analysis techniques aim to identify security threats to software systems and to plan countermeasures to mitigate them. Yet, empirical evidence shows that existing threat analysis techniques can be manually labor intensive and lack in automation. Furthermore, design-level models are seldom kept in sync with the implementation, potentially resulting in architectural erosion and technical debt. Threat analysis is often performed on *Data Flow Diagrams* (DFD), an informal representation of the software architecture. To support the detection of problematic information flows, in our rearlier work we introduced *SecDFD*, an extension of the DFD notation supporting the specification of security-relevant information [TBS19]. However, the outcomes of such

[1] University of Koblenz-Landau, Koblenz, Germany, EMail: speldszus@uni-koblenz.de, juerjens@uni-koblenz.de
[2] University of Gothenburg and Chalmers University of Technology, Gothenburg, Sweden, EMail: katja.tuma@cse.gu.se, danstru@chalmers.se, riccardo.scandariato@cse.gu.se
[3] Fraunhofer Institute for Software and Systems Engineering, Dortmund, Germany

doi:10.18420/SE2020_13

detection are of limited value if the implementation does not comply with the security properties described in the DFD model.

Our work aims to support the discovery of secure data-flow compliance violations between the designed and the implemented system. We present a technique that automatically establishes mappings between a design-level model enriched with security-relevant information (SecDFD) and an implementation-level model (*Program Model* [Pe15]). These mappings can be used to discover compliance violations of secure data-flow properties: First, the designed data flow is captured in the SecDFD model and afterwards the actual data flow is obtained from implementation-level data-flow analysis tools. These tools typically require sophisticated meta-data (e.g. an explicit tagging of security-critical data) as input, which can be obtained from our mappings. Finally, our mappings also support the designer in an early discovery of implementation absence, convergence, and divergence with respect to the planned software design and its security properties. We make the following contributions:

(i) We present an automated technique for establishing mappings between SecDFDs and program models, thereby supporting the discovery of secure data-flow compliance violations. The key idea of our technique is twofold. First, we define a mapping between SecDFD and program-model element types, constraining how elements can be mapped to each other. Second, we combine similarity-based matching of element names with structural heuristics (based on data-flow properties) to automatically derive suggested mappings between the SecDFD and the program model.

(ii) We present an incremental methodology, in which the user is involved to successively discover new mappings and eventually derive an adequate mapping.

(iii) We present our implementation of the approach as a publicly available Eclipse plugin and the evaluation of its accuracy on five open source Java projects.

Our tool implementation as well as all experimental data sets are available on our GitHub site (`https://github.com/SvenPeldszus/GRaViTY-SecDFD-Mapping`).

References

[Pe15] Peldszus, S.; Kulcsár, G.; Lochau, M.; Schulze, S.: Incremental Co-Evolution of Java Programs based on Bidirectional Graph Transformation. In: PPPJ. 2015.

[Pe19] Peldszus, S.; Tuma, K.; Strüber, D.; Jürjens, J.; Scandariato, R.: Secure Data-Flow Compliance Checks between Models and Code based on Automated Mappings. In: MODELS. 2019.

[TBS19] Tuma, K.; Balliu, M.; Scandariato, R.: Flaws in Flows: Unveiling Design Flaws via Information Flow Analysis. In: ICSA. 2019.