Integrating Security-Enriched Data Flow Diagrams Into Architecture-Based Confidentiality Analysis

Nils Niehues
nils.niehues@kit.edu
Karlsruhe Institute of Technology (KIT)

Benjamin Arp
benjamin.arp@student.kit.edu
Karlsruhe Institute of Technology (KIT)

Tom Hüller
tom.hueller@student.kit.edu
Karlsruhe Institute of Technology (KIT)

Felix Schwickerath
felix.schwickerath@student.kit.edu
Karlsruhe Institute of Technology (KIT)

Nicolas Boltz
nicolas.boltz@kit.edu
Karlsruhe Institute of Technology (KIT)

Sebastian Hahner
sebastian.hahner@kit.edu
Karlsruhe Institute of Technology (KIT)

Abstract

The increasing complexity of modern software systems presents developers with significant challenges regarding the confidentiality of sensitive data. To this end, data flow diagrams serve as an effective tool for identifying potential confidentiality violations. Previous work in this area collected a data set comprising security-enriched data flow diagrams. Previous work on the security of microservice applications has created an extensive dataset of security-enriched data flow diagrams derived from open-source projects. The data set also includes security rules for microservices architectures specified in natural language. This paper presents an automated pipeline that converts descriptions of data flow diagrams with security rules into models suitable for automated information security analysis. Our evaluation based on the existing data set shows that the transformed models are highly accurate, establishing a gold standard for data flow-based confidentiality analysis.

1 Introduction

In today's digital world, safeguarding sensitive data is more crucial than ever. As software systems become increasingly complex, protecting this data presents significant challenges for developers. Ensuring that information remains secure from potential threats is vital for maintaining trust and ensuring the smooth operation of these systems. The value of this data is intrinsically linked to its confidentiality; if it cannot be trusted, its usefulness is greatly diminished.

In this context, data flow diagrams serve as an effective tool for identifying potential confidentiality violations. To support this, Schneider et al. [4] introduced microSecEnD, a comprehensive dataset that features security-enriched data flow diagrams derived from real-world open-source microservice applications. The dataset further includes security rules for microservice architectures written in natural language. It also contains variant models, each complying with a specific rule.

This paper presents an automated pipeline that operates on the microSecEnD repository shown in Figure 1. In Section 2 we transform the microSecEnD dataset into models suitable for confidentiality analysis. We then formalize the natural language security rules in predicate logic in Section 3.

We evaluate our pipeline and the resulting dataset in Section 4, demonstrating high accuracy in both the transformation and the formalization of constraints. Using this process, we create an evaluated *gold standard*, i.e., a reference for future confidentiality analysis benchmarks. We aim to provide a baseline that future researchers and practitioners can use to assess and improve their own confidentiality analysis approaches.

2 Transformation

The microSecEnD dataset [4] consists of 17 different open-source microservice applications, each represented as a data flow diagram with annotated nodes and flows. These annotations include details such as whether a node is *internal* or if a connection represented by a flow is *RESTful*. Additionally, the authors have provided variants for each base model, with each variant following a specific security rule.

Label propagation is a widely utilized technique for assessing data flow diagrams for potential confidentiality violations [2]. This method involves assigning labels to both the nodes and the data flowing between them and then propagating these labels along the flow of data according to the propagation behavior defined for each node individually. After propagation, the labels present at each node can be checked to identify confidentiality violations.

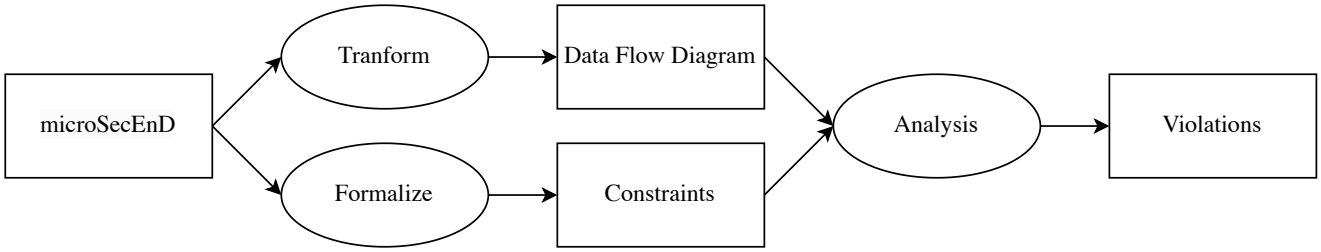

Figure 1: Pipeline from microSecEnD dataset to confidentiality violations

To conduct our confidentiality analysis, data flow diagrams in microSecEnD must be converted into a suitable format, which involves several key steps:

First, we address label propagation by ensuring the model forms a directed acyclic graph (DAG). This requires unrolling cycles in the models to process each cycle only once, thereby more accurately preserving the intended behavior. It is important to note that models composed entirely of cycles cannot be unrolled, as no distinct starting point can be identified.

Next, we generate labels by utilizing the node and flow annotations of the microSecEnD dataset to create node labels and labels for each node's outgoing data. The propagation behavior of the nodes is then configured to forward incoming data while appending their node and outgoing data labels. This forwarding mechanism, while approximating the actual system behavior, is essential for accurately tracking the path of data through the data flow diagram, enabling us to detect any potential confidentiality violations effectively.

The final output of our transformation is a cycle-free data flow diagram based on the DFD metamodel of Boltz et al. [3], which includes all derived security-relevant labels and behaviors.

3 Constraints

Schneider et al. [4] define different security rules in their paper and provide variants designed to adhere to a specific rule. These rules are defined in natural language, such as

> All communication traffic from external users and entities should be encrypted using secure communication protocols.

Because these rules are provided in natural language, they cannot directly be used for automated analyses. We formalize our constraints based on node labels and incoming data labels to address this. These labels result from the transformation in Section 2 and the label propagation of the confidentiality analysis [3]. We define the following predicates for our constraints:

$N(x)$ True if x is a node

$NL(x, y)$ True if node x has node label y

$DL(x, y)$ True if node x has data label y

Based on these predicates, the above example can be formalized as follows:

$$\neg\exists x \left(N(x) \wedge DL(x, \text{entrypoint}) \wedge \neg DL(x, \text{encrypted})\right)$$

This can be interpreted as no node shall receive unencrypted data from an entry point, which the analysis can automatically check.

From the 18 different rules Schneider et al. [4] defined, we purposefully exclude rules 13 to 17 as these deal with the qualitative goals of systems rather than security goals. An example being

> The API gateway should perform load balancing.

We formalize the remaining 13 rules following our notation, and Table 1 shows the constraints in predicate logic paired with their IDs from [4].

4 Evaluation

Following the guidelines of Konersmann et al. [1], we evaluate the accuracy of the microSecEnD dataset, our transformation in data flow diagrams, and our formalized constraints to establish a gold standard for confidentiality analysis.

microSecEnD contains 17 base models with up to 18 variant models for specific rules. We exclude the variants whose rules concern qualitative attributes rather than security. As mentioned before, complete cyclic models cannot be analyzed and must be excluded, leaving us with **132** variant models. We then run our pipeline to feed the transformed models and constraints into our confidentiality analysis and observe which model variants break which constraints.

We define a transformation and constraint as *accurate* if a constraint never flags a violation on a transformed model variant that was specifically designed to satisfy that constraint. After running the confidentiality analysis, **115/132** model variants met this criterion, resulting in an initial accuracy of **0.87**.

Upon closely inspecting the 17 model variants that did not behave as expected, we discovered that the issue was not with the transformation, or our formalization of the constraints but rather with inherent faults in the variants from microSecEnD. For instance, some

ID	Constraint
1	$\neg\exists x\,(N(x) \wedge (NL(x, \text{internal}) \wedge DL(x, \text{entrypoint}) \wedge \neg DL(x, \text{gateway})) \vee (NL(x, \text{gateway}) \wedge NL(\text{internal})))$
2	$\neg\exists x\,(N(x) \wedge DL(x, \text{internal}) \wedge \neg DL(x, \text{authenticated}))$
3	$(\exists x\,(N(x) \wedge NL(x, \text{auth server})) \wedge (\neg\exists y\,(N(y) \wedge NL(y, \text{internal}) \wedge DL(y, \text{entrypoint}) \wedge \neg DL(y, \text{auth server})))$
4	$\neg\exists x\,(N(x) \wedge NL(x, \text{internal}) \wedge DL(x, \text{entrypoint}) \wedge \neg DL(x, \text{anonymized}))$
5	$\neg\exists x\,(N(x) \wedge DL(x, \text{entrypoint}) \wedge NL(x, \text{internal}) \wedge \neg NL(x, \text{validated}))$
6	$(\exists x\,(N(x) \wedge NL(x, \text{auth server})) \wedge (\neg\exists y\,(N(y) \wedge NL(y, \text{auth server }) \wedge \neg NL(y, \text{regulated})))$
7	$\neg\exists x\,(N(x) \wedge DL(x, \text{entrypoint}) \wedge \neg DL(x, \text{encrypted}))$
8	$\neg\exists x\,(N(x) \wedge DL(x, \text{internal}) \wedge \neg DL(x, \text{encrypted}))$
9	$\exists x\,(N(x) \wedge NL(x, \text{logging server})$
10	$\neg\exists x\,(N(x) \wedge NL(x, \text{internal}) \wedge \neg NL(x, \text{logged process}))$
11	$\neg\exists x\,(N(x) \wedge (NL(x, \text{internal}) \wedge \neg NL(x, \text{logged process})) \vee (NL(x, \text{logged process}) \wedge \neg NL(x, \text{sanitized})))$
12	$(\exists x\,(N(x) \wedge NL(x, \text{logging server})) \wedge (\neg\exists y\,(N(y) \wedge NL(y, \text{logging server}) \wedge \neg DL(y, \text{message broker})))$
18	$\exists x\,(N(x) \wedge NL(x, \text{secret manager})$

Table 1: Formalized constraints

data flow diagrams allowed unauthorized data flows to internal nodes by bypassing the authorization server, leading to constraint violations. After excluding these faulty variants, we raised the accuracy to **1.0**, establishing a gold standard consisting of **115** models.

To address the threat to internal validity posed by the assessment of the faulty models, we mitigated this by having three researchers independently identify the faults. All researchers consistently identified the same problematic parts of the models. A potential threat to external validity is that the final set of **115** models might not be fully representative; however, this risk is mitigated by the diverse backgrounds of the microservice applications and security rules used. To enhance reliability, we provide our pipeline and dataset [5], which are also available online at `https://dataflowanalysis.org/`.

5 Conclusion

This paper presents an automated pipeline that transforms the microSecEnD dataset into models optimized for rigorous confidentiality analysis. By formalizing natural language security rules into predicate logic, we ensured precise detection of potential confidentiality violations. Our evaluation confirmed the high accuracy of the transformed models and constraints, establishing a robust gold standard for data-flow-based confidentiality analysis.

Looking ahead, we plan to collaborate with Schneider et al. [4] to manually repair the identified faulty models, enhancing the dataset's overall integrity. Additionally, we aim to explore methods for effectively analyzing models that are entirely cyclic, which currently pose challenges due to the lack of a clear starting point.

We intend to apply this gold standard in various research areas, including confidentiality analysis under uncertainty, automatic model repair, data protection analysis, and developing recommender systems for system architects. These efforts will further refine and expand the utility of our approach in securing complex software systems.

Acknowledgements

This publication is partially based on the research project SofDCar (19S21002), which is funded by the German Federal Ministry for Economic Affairs and Climate Action. This work was also supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs, the BMBF (German Federal Ministry of Education and Research) grant number 16KISA086 (ANYMOS), and the NextGenerationEU project by the European Union (EU).

References

[1] M. Konersmann et al. "Evaluation Methods and Replicability of Software Architecture Research Objects". In: *2022 IEEE 19th International Conference on Software Architecture (ICSA)*. 2022, pp. 157–168.

[2] S. Seifermann et al. "Detecting Violations of Access Control and Information Flow Policies in Data Flow Diagrams". In: *The journal of systems and software* 184 (2022). 46.23.03; LK 01, Art.–Nr. 111138.

[3] N. Boltz et al. "An Extensible Framework for Architecture-Based Data Flow Analysis for Information Security". In: *European Conference on Software Architecture*. Springer. 2023, pp. 342–358.

[4] S. Schneider et al. "microSecEnD: A Dataset of Security-Enriched Dataflow Diagrams for Microservice Applications". In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 125–129.

[5] N. Niehues et al. *Supplementary Material for "Integrating Security-Enriched Data Flow Diagrams Into Architecture-Based Confidentiality Analysis"*. 2024. URL: `https://doi.org/10.5281/zenodo.13382614`.