

Quantitative Information Flow Control by Construction for Component-Based Systems

Rasmus C. Rønneberg

Karlsruhe Institute of Technology, Germany
rasmus.ronneberg@kit.edu

Abstract. Secure software architecture is increasingly important in a data-driven world. When security is neglected sensitive information might leak through unauthorized access. To mitigate this software architects needs tools and methods to quantify security risks in complex systems. This paper presents doctoral research in its early stages concerned with creating constructive methods for building secure component-based systems from a quantitative information flow specification. This research aim at developing a method that allows software architects to develop secure systems from a repository of secure components. Planned contributions are refinement rules for secure development of components from a specification and well-formedness rules for secure composition of said components.

Keywords: Information flow control · Component-based systems · Secure by construction

1 Research problem

Systems today operate on large amounts of data. This creates value for society, both in terms of scientific and commercial value. However, we must ensure the *confidentiality* of that data, such that systems do not leak sensitive information through unauthorized access. And we must consider the *integrity* of systems to ensure that they function correctly and are not influenced by untrusted actors. These challenges are especially prominent in security-critical software that handles sensitive information about individuals, such as location data in the mobility domain. A key challenge is to provide software architects with methods that allow them to build these security-critical systems and reason about security in large and complex systems on an architectural level. One framework for analyzing security is information flow control (IFC) using a property such as non-interference [8]. Non-interference can be used to analyze whether a specified security policy has been breached, and provide an yes/no answer to this question. For instance information flow control has been successfully used to statically check security breaches in programs through types systems [12,14]. Non-interference has also been used as the formalism for security in component-based software architectures for cyber-physical systems [3]. These types of work enable information flow control at the component and architectural level in a post-hoc analysis. We propose building secure systems in an incremental way analogous to the

correctness-by-construction (CbC) [7] approach for functional correctness. CbC has been raised to the architectural level by enabling correctness-by-construction for component-based system in ArchiCorC [6]. However, ArchiCorC only considers functional correctness and not security and information flow control. This work aim at closing the gap between CbC and information flow control at the architectural level. An additional challenge is that the previously mentioned work does not support quantification of information flow. Approaches such as quantitative information flow [1] aim at quantifying exactly how secure or insecure systems are. Quantification of information flow is generally used in scenarios where some amount of information leakage in a program is allowed, and standard information flow would be too restrictive. Currently, CbC approaches can not be used to build systems with quantitative security guarantees. In this work we want to build software components that are secure by construction, guided by a quantitative information flow security specification, because this yields strong guarantees for the specification in the resulting system, and we want to study which security guarantees we can establish when these components are composed in different ways. With these results, we want to build and scale a unified way of constructing secure software from the architectural level down to the source code. In summary the research questions are as follows:

Main research question: How can we define an incremental approach to constructing secure systems from secure components that adhere to a quantitative information flow policy and ensure security by construction?

This leads to the following subquestions guiding the research methodology.

RQ1: How can we devise a constructive approach to building secure software components complying to a quantitative information flow policy?
RQ2: How can we compose these components such that the overall system adheres to the quantitative security specification?
RQ3: How can we use such an approach to building secure components with different execution and computational models?

2 State of the art

The work related to the above research questions ranges from component-based secure architectures, to the construction of secure components, and finally to quantification of security. There exist several methods for constructing and analyzing secure component-based architectures using different computational models such as timed automata and labeled transitions systems [3,9,4,5]. These methods are not constructive and they use non-interference as the formalism of security. In contrast, our approach is constructive and uses quantitative information flow as the formalism for security.

Approaches to building secure software components include post-hoc analysis using both static and dynamic methods and also CbC approaches. In most post-hoc methods security is ensured through a security type system or a monitor at

runtime [12,14,2]. CbC approaches such as in the work by Runge et al. [11,10] aim at constructing secure programs by refinements of specifications. However, these approaches do not scale to the architectural level, and they do not study the composition of components. Furthermore, they do not consider quantification of information flow.

Quantitative information flow has been applied to analyse information leakage in programs and as a hyperproperty on the execution traces of programs [15,13]. But crucially, these works do not consider quantitative information flow as a security specification and do not derive programs through refinements, as this work proposes. Furthermore, they also do not study the compositional capabilities of quantitative information flow.

3 Quantitative information flow control by construction

The overarching idea of the information flow control approach is to allow software architects to perform a high-level component-based design of the system. Components are functionally connected through provided and required interfaces. Each component is secure-by-construction. The collection of secure components forms a repository that can be used and reused in different settings. An overview of the envisioned development process is shown in Figure 1. As shown in step

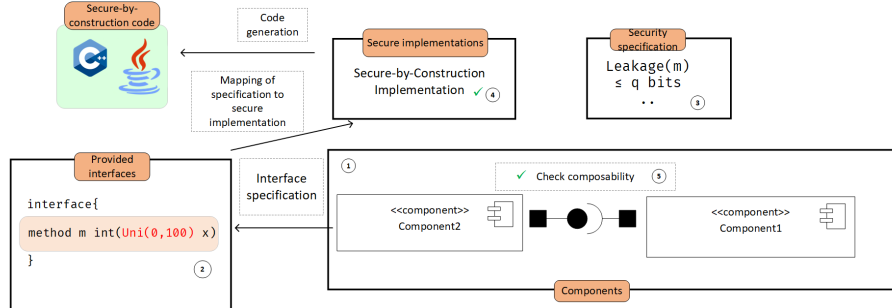


Fig. 1. Quantitative Information Flow Control Development Process

① we envision that components are connected with other components through required and provided interfaces. We assume a component model with composite and basic components. Composite components can contain subcomponents. Basic components are mapped to the provided interface of said component. ② A provided interface is a set of method signatures. The signatures specify a name, return type, and parameters of the method. Furthermore, they will also contain a specification of the distributions that parameters are assumed to be sampled from, which is necessary for the quantitative information flow control. ③ All

methods in the provided interfaces have a security specification. The security specification defines an upper bound on how much information can leak from the input to the output of the method. The upper bounds are specified in terms of quantitative information flow metrics e.g. *Shannon-entropy*, *guessing-entropy*, and *min-entropy* [1]. ④ The methods in the provided interfaces are mapped to secure implementation. The implementations are constructed incrementally through sound refinement rules. For each mapping we check that the implementation adheres to the security specification. ⑤ When components are composed, we check that the composition does not violate any of the security specifications and that the overall system is still secure. Finally our approach allows to generate code that is secure by construction.

4 Research methodology and evaluation

To realize the proposed solution, we need a mixture of different research methods. The first step is a *systematic literature review* of state of the art for secure software architectures. The second step is to develop the *formal theory* for deriving and composing components from a quantitative information flow policy. The third step is to build a prototype and to evaluate it using a *case study*. The fourth step is to do an *empirical evaluation* to understand whether the proposed solution can help experts in the field with developing security-critical systems.

We plan to evaluate our approach on real-life case studies in the mobility domain, where we intend to use our tool to develop software components for an autonomous shuttle operating in a ridepooling setting. Furthermore, we plan to do expert interviews with software architects and security experts to gather feedback from people in industry. The expected results of this work are: (1) A set of refinement rules for incremental and secure development of components according to a quantitative security specification. (2) Well-formedness rules for secure composition of components. (3) Tool support for developing security-critical software based on the developed concepts.

Possible limitation of this work that we foresee are a trade-off between usefulness and soundness. In some cases, we might need to choose between expressiveness and the ability to prove security guarantees. If successful, this work will bring together correctness-by-construction, secure software architecture, and quantitative information flow.

Acknowledgments This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs

References

1. Alvim, M.S., Andrés, M.E., Palamidessi, C.: Probabilistic Information Flow. In: 2010 25th Annual IEEE Symposium on Logic in Computer Science. pp. 314–321. IEEE, Edinburgh, United Kingdom (Jul 2010)

2. Asplund, M., Nadjm-Tehrani, S. (eds.): Secure IT Systems: 25th Nordic Conference, NordSec 2020, Virtual Event, November 23–24, 2020, Proceedings, Lecture Notes in Computer Science, vol. 12556. Springer International Publishing, Cham (2021)
3. Gerking, C., Schubert, D.: Component-Based Refinement and Verification of Information-Flow Security Policies for Cyber-Physical Microservice Architectures. In: 2019 IEEE International Conference on Software Architecture (ICSA). pp. 61–70. IEEE, Hamburg, Germany (Mar 2019)
4. Greiner, S., Grahl, D.: Non-interference with What-Declassification in Component-Based Systems. In: 2016 IEEE 29th Computer Security Foundations Symposium (CSF). pp. 253–267. IEEE, Lisbon (Jun 2016)
5. Jasser, S., Tuma, K., Scandariato, R., Riebisch, M.: Back to the Drawing Board - Bringing Security Constraints in an Architecture-centric Software Development Process. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. pp. 438–446. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal (2018)
6. Knüppel, A., Runge, T., Schaefer, I.: Scaling Correctness-by-Construction. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles, vol. 12476, pp. 187–207. Springer International Publishing, Cham (2020), series Title: Lecture Notes in Computer Science
7. Kourie, D.G., Watson, B.W.: The Correctness-by-Construction Approach to Programming. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
8. Mantel, H.: Information Flow and Noninterference. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security, pp. 605–607. Springer US, Boston, MA (2011)
9. Mohammad, M., Alagar, V.: A formal approach for the specification and verification of trustworthy component-based systems. *Journal of Systems and Software* **84**(1), 77–104 (Jan 2011)
10. Runge, T., Knüppel, A., Thüm, T., Schaefer, I.: Lattice-Based Information Flow Control-by-Construction for Security-by-Design. In: Proceedings of the 8th International Conference on Formal Methods in Software Engineering. pp. 44–54. ACM, Seoul Republic of Korea (Oct 2020)
11. Runge, T., Servetto, M., Potanin, A., Schaefer, I.: Immutability and Encapsulation for Sound OO Information Flow Control. *ACM Transactions on Programming Languages and Systems* **45**(1), 1–35 (Mar 2023)
12. Sabelfeld, A., Myers, A.: Language-based information-flow security. *IEEE Journal on Selected Areas in Communications* **21**(1), 5–19 (Jan 2003)
13. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: Proceedings 13th IEEE Computer Security Foundations Workshop. CSFW-13. pp. 200–214. IEEE Comput. Soc, Cambridge, UK (2000)
14. Volpano, D., Irvine, C., Smith, G.: A sound type system for secure flow analysis. *Journal of Computer Security* **4**(2-3), 167–187 (Apr 1996)
15. Yasuoka, H., Terauchi, T.: Quantitative information flow as safety and liveness hyperproperties. *Theoretical Computer Science* **538**, 167–182 (Jun 2014)