Security Guarantees and Evolution: From Models to Reality*

Martín Ochoa[1,2]

[1] Siemens AG, Germany
[2] TU Dortmund, Germany
martin.ochoa@cs.tu-dortmund.de

Abstract. Achieving security in practical systems is a hard task. As it is the case for other critical system properties (i.e. safety), security should be a concern through all the phases of software development, starting with the very early phases of requirements and design. Although the arguments in favour of formal verification at the design level are many (rigour and cost-benefit being at the top of the list), answers to two relevant questions about the limitations of this approach are crucial for its success and further application in industrial contexts: Is security verification at the design phase scalable under continuous evolution? What are the limits of the formal security guarantees achieved at the model level when software is ultimately deployed? In this paper we report on recent results and work in progress towards a better understanding of these two fundamental questions.

1 Motivation

In recent years, information security has gained increasing attention from the general public and there is a consensus about its paramount importance in society. Examples include recent scandals on users private data [11], leaks of government secret documents and public threats from anonymous hacker groups to corporate and governmental IT systems worldwide [1,2]. Long gone are the days where the term 'computer security' was associated exclusively with spies, conspirational theories and cryptography. Today most successful attacks exploit vulnerabilities related to problems in design or implementation rather than vulnerabilities in cryptographic mechanisms. And most of the attackers are motivated teenagers, typically not interested in the mathematical aspects of cryptography.[3]

There are several reasons why security is difficult to achieve in practice. On the one hand, the complexity of modern system architectures is constantly increasing: software logic evolves, often driven by the market pressure to deliver new functionalities, and different operating systems and hardware configurations

* PhD work supervised by Jan Jürjens at the Chair for Software Engineering of the TU Dortmund and Jorge Cuéllar at Siemens Corporate Research.

[3] Also social aspects of security result in attacks in many cases, but those are very difficult to control by technological means and are out of the scope of this work.

make each instantiation of an IT system unique. Moreover, software components from different producers delivered without accurate (if at all) security guarantees are used together to achieve customized solutions.

Techniques to tackle this ever 'moving target' exist in different areas of computer science and engineering: from software and hardware formal verification to testing, but also at the level of business-processes modelling and risk-analysis. In fact, security plays a role at different levels of abstraction and at different phases of the development cycle, and if one wants to have a high degree of assurance about the security of a system one should consider them all.

In general, the strongest guarantees about software and hardware behaviour are delivered by formal methods, due to their rigour and precision. It is thus natural to consider formal methods to validate a system with respect to well-defined, mathematical descriptions of security. Nevertheless, formal methods are difficult to apply in realistic software-development scenarios because they require highly specialized designers and programmers in order to carry out the formalizations and because many verification tools available hardly scale to realistic scenarios, in particular at the level of implementations. Additionally, the security requirements of a system are not always precise, because often they are formulated in natural language.

At the present time, the application of formal methods seems to be more reasonable to take place at the level of system design, where models are abstracted from implementation details and are more amenable to automated verification. Since some security problems can be detected already at the specification level, the cost-benefit of applying this methodology is typically better than repairing design problems at later stages of development. Since the de-facto standard for system modelling in industry is the Universal Modelling Language (UML), it is natural to perform this formal verification on UML models, if one aims at industry acceptance. It is however not enough to have strong security guarantees on system models to be able to judge the overall security of a deployed system, which is the ultimate goal. As argued before, it is difficult to verify the code because of its size and the huge variety of programming languages and paradigms. Moreover it can be the case that libraries or components from third parties are used whose source code is unavailable.

Unfortunately, security is a never ending open loop, since no matter how strong guarantees a given system has, new exploits appear that consider properties or interfaces that were not considered at design. This is prominently illustrated by side-channels: here the attacker exploits information such as electromagnetic radiation, timing and shared memory behaviour to gain possession of confidential data. Many of these side-channels are difficult to capture since they rely on micro-architectural configurations such as the duration of processor instructions or the cache behaviour. Closing these interfaces is sometimes impossible or costly, because of the impact to other system requirements (i.e. efficiency).

In summary, we believe that it is unrealistic to attempt to build complex industrial systems with a 100% guarantee of security because logical or physical

interfaces that are vulnerable to attacks are often only exposed after real attacks take place. To date there is also no standard single methodology, tool, or model to tackle the whole Software Development Life-Cycle. It is thus necessary to provide tools that help to cope with evolution problems at all levels of abstractions and during the whole life-cycle. It is our believe that the idea that systems are going to be secure because we commit to a single approach (for example rigorous security requirements elicitation, industrial best-practices, strict patch update policies etc.) is fallacious: we have to work on all phases and at different abstraction levels, sometimes using different models and different methodologies. In this work we propose a set of such tools, easy to use by end users, and addressing the aforementioned problems on Models, Implementations and Micro-architectures.

Fig. 1. Roadmap from models to running systems

2 Roadmap and contributions

In the following we summarize the goals of this research and the achieved results so far. Security is typically described as the conjunction of one or more security requirements, abstractly classified as Confidentiality, Availability and Integrity. In this work we consider mainly Confidentiality and Integrity: we want to understand how information is flowing from a group of users to another. There are basically two ways to look at the problem and our proposed solutions: a) according to the abstraction level modelled and b) according to the security properties considered. First, we will take the first point of view (see Fig. 1).

Models We consider the problem of specification evolution for security at the level of UML models. We extend the UMLsec [5] notation and verification techniques to reason about changes in the specification by means of the novel UMLseCh notation [7,6]. For some properties defined in static UML diagrams, we describe sufficient conditions that soundly preserve the security of already verified models by analysing the delta implied by the modifications. This fine-grained incremental technique is a good choice for structural of properties because of their locality: changes of parts of the model usually affect the property in a clearly identifiable, small subset of the specification. For behavioural models, incremental changes can affect security in non-trivial ways. Therefore we propose to focus on compositionality results: if one or more components of a given system evolves, the overall security of the system can be decided (efficiently) by re-verifying the evolved components exclusively given an (efficient)

compositionality condition. We describe such a decision procedure for secrecy on cryptographic protocols specified as Sequence diagrams [10] that is sound and complete with respect to a previous verification technique proposed by Jürjens [5]. We currently extend previous work on verifying non-interference in UML state-charts and derive compositionality results for interacting objects.

Implementations For evaluating the security of implementations we consider model-based testing of security requirements based on a black box model of the system. The methodology proposed is well-founded with respect to the requirements considered, and extends previous work on security testing [4]. We also discuss conditions under which the security relevant properties of the model are preserved under incremental changes [3].

Micro-architecture At this detailed abstraction level, we focus on cache configurations and how they can act as a leaking channel for different adversaries. Configurations of the CPU play a determinant role on the security of the system with respect to side-channel attacks, and the change in configurations is a typical phenomenon of system's evolution. Avoiding the use of caches conflicts with efficiency requirements and is therefore not realistic for a wide range of systems. A promising technique to achieve formal guarantees about countermeasures striving for a trade-off between security and other conflicting requirements is quantitative information flow analysis (for example [8]). We formalize heuristic countermeasures proposed in the literature and give strong security guarantees for arbitrary programs under a well-defined attacker model [9], and validate our approach using an automatic tool chain that evaluates compiled programs for various architectures.

Notice that we do not aim at a formal integration of the security guarantees obtained at the discussed levels of abstraction: such a task, although interesting, goes outside the scope of this work. On the other hand, current industrial environments do not have a formally justified software development process. We consider different layers of abstraction mainly due to necessity: any error found in any of these layers would invalidate the over-all security of the system.

It is nevertheless interesting to informally discuss our contributions from the perspective of the security properties considered and the assumptions they rely on at different abstraction levels. In fact, when analysing information-flow at the model level, we are assuming perfect mechanisms for access control, and among others, perfect cryptography, which guarantees access control when information is shared through insecure networks. To gain confidence in the correctness of those mechanisms, we validate them locally using model-based testing and performing a Dolev-Yao analysis for the cryptographic protocol logic. To gain even more confidence about primitives, we consider the micro-architectural abstraction level, using quantitative information-flow related techniques. We can see this as an (informal) chain of assumptions and guarantees across abstraction levels, as depicted in Fig. 2.

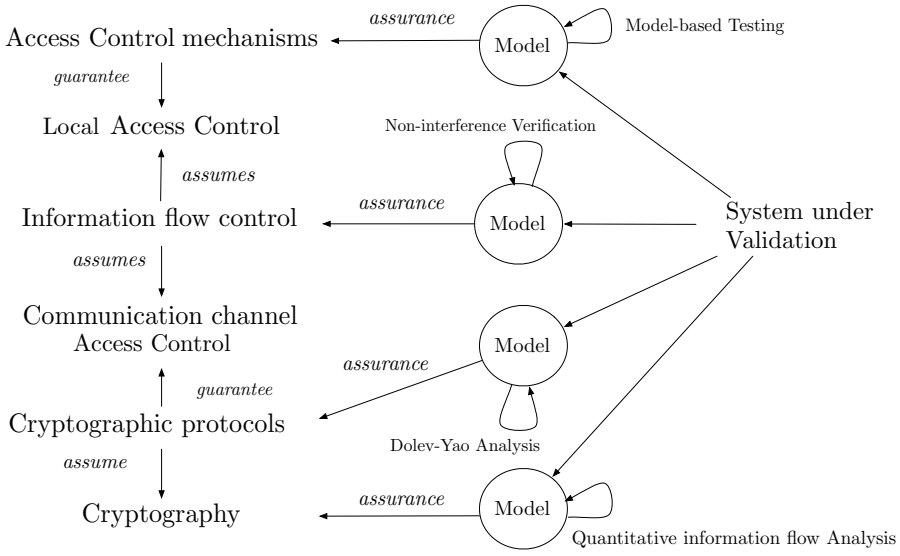

Fig. 2. Access control and Information-flow control vs. model validation

The assumptions and guarantees discussed here are by no means complete: for example we are not considering operating system access control mechanisms or semantic security properties of the cryptographic primitives. As already discussed we believe that a complete and formally justified methodology is unrealistic. It is nevertheless essential to consider different levels of abstraction and development phases to achieve a good degree of confidence in the system, since an error in any of them would invalidate the results at higher abstraction levels or previous phases. For example a faulty implementation of access control would invalidate a secure abstract design w.r.t non-interference, and a cryptographic implementation with side-channels would make a formally verified protocol against the Dolev-Yao adversary meaningless.

3 Conclusions and Work in Progress

Software security is a difficult problem because it is a moving target, and it should be addressed at different levels of abstraction and in all phases of software development. Although a promising methodology to achieve practical security is formal verification at design time, to date there a number of limitations to this approach, in particular when the system undergoes continuous evolution. We have reported on results towards a scalable security verification at the model level, and pinned down many assumptions made at different levels of abstractions, that are vital for the formal model analysis to be sound. Our focus is to devise methodologies supporting intuitive tools, aiming at minding the gap between formal methods and industrial acceptance. At this point of our research,

we have already achieved many of the original goals, and promising work in progress is being done on the missing items. Concretely, at this stage work in progress is being done for information flow analysis: at the model level, we are interested in the automatic and efficient verification of UML state-charts; at the micro-architectural level novel formal quantification techniques are being studied that provide strong security guarantees on realistic modern processor models for powerful attackers and multiple hardware configurations. Submissions to international conferences in software engineering and automatic verification in both of these subjects are on preparation at the time of writing this manuscript.

Acknowledgements This research was partially supported by the MoDelSec Project of the DFG Priority Programme 1496 "'Reliably Secure Software Systems – RS^3" and the EU project NESSoS (FP7 256890).

References

1. LulzSec hackers claim CIA website shutdown. BBC news `http://www.bbc.co.uk/news/technology-13787229`, 2011. Online, accessed 04-Feb-2012.
2. LulzSec takes down Brazil government sites. Cnet news `http://news.cnet.com/8301-1009_3-20073219-83/lulzsec-takes-down-brazil-government-sites/`, 2011. Online, accessed 04-Feb-2012.
3. E. Fourneret, F. Bouquet, M. Ochoa, J. Jürjens, and S. Wenzel. Vérification et test pour des systèmes évolutifs. In *Approches Formelles dans l'Assistance au Dveloppement de Logiciels (AFADL)*, 2012.
4. E. Fourneret, M. Ochoa, F. Bouquet, J. Botella, J. Jürjens, and P. Yousefi. Model-based security verification and testing for smart-cards. In *Proceedings of the 6th International Conference on Availability, Reliability and Security (ARES)*, pages 272–279. IEEE, 2011.
5. J. Jürjens. *Secure Systems Development with UML*. Springer, 2005.
6. J. Jürjens, L. Marchal, M. Ochoa, and H. Schmidt. Incremental Security Verification for Evolving UMLsec models. In *Proceedings of the 7th European Conference on Modelling Foundations and Applications (ECMFA)*, volume 6698 of *Lecture Notes in Computer Science*, pages 52–68. Springer, 2011.
7. J. Jürjens, M. Ochoa, H. Schmidt, L. Marchal, S. H. Houmb, and S. Islam. Modelling secure systems evolution: Abstract and concrete change specifications. In *Proceedings of the 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*, volume 6659 of *Lecture Notes in Computer Science*, pages 504–526. Springer, 2011.
8. B. Köpf and M. Dürmuth. A provably secure and efficient countermeasure against timing attacks. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium (CSF)*, pages 324–335. IEEE Computer Society, 2009.
9. B. Köpf, L. Mauborgne, and M. Ochoa. Automatic quantification of cache side-channels. Cryptology ePrint Archive, Report 2012/034, 2012. `http://eprint.iacr.org/`.
10. M. Ochoa, J. Jürjens, and D. Warzecha. A sound decision procedure for the compositionality of secrecy. In *Proceedings of the 4th International Symposium on Engineering Secure Software and Systems (ESSoS)*, 2012.
11. M. J. Schwartz. Sony sued over playstation network hack. Information Week, `http://www.informationweek.com/news/security/attacks/229402362`, 2011. Online, accessed 04-Feb-2012.