SecureChange: Security Engineering for Lifelong Evolvable Systems

Riccardo Scandariato[1] and Fabio Massacci[2]

[1] IBBT-DistriNet
Katholieke Universiteit Leuven
3001 Leuven, Belgium
[2] Dipartimento di Ingegneria e Scienza dell'Informazione
Università di Trento
38050 Trento, Italy

Abstract. The challenge of SecureChange is to re-invent security engineering for "eternal" systems. The project focuses on methods, techniques and tools to make change a first-class citizen in the software development process so that security is built into software-based systems in a resilient and certifiable way. Solving this challenge requires a significant re-thinking of all development phases, from requirements engineering to design and testing.

Keywords: Security, evolvability, software life-cycle.

1 Introduction

A reality-check on complex critical systems reveals that their life-cycle is characterized by a short design time (months) compared to the much longer operational time (many years or even decades). Due to these considerations, it is unlikely that such systems will remain unchanged during their life time. Hence, a primary need for those software-based systems is to be highly flexible and evolvable. At the same time, those systems expose strong security and dependability requirements as they handle sensitive data, impact our daily life, or put people's lives at stake. Current software engineering techniques are not equipped to deal with this conflicting situation. On one side, state-of-the-art research has unearthed methods and techniques to support *adaptability*. In this respect, autonomic systems represent, for instance, one of the most promising directions. However, the high flexibility of adaptable systems comes with the cost of little or no guarantees that the security properties are preserved with change. On the opposite side, very precise *verification* techniques work under the assumption that the properties to-be-verified and the system under verification are fixed over time. That is, they are conceived for rigid, inflexible systems.

The challenge of SecureChange is to re-invent security engineering for "eternal" systems. The project focusses on methods, techniques and tools to make *change* a first-class citizen in the software development process so that security is built and certified in software-based systems in a way that is resilient to change. Solving the above-mentioned challenge requires a significant re-thinking of all development phases, from requirements engineering to design and testing.

2 The Project at a Glance

All the activities carried out by the project consortium revolve around a common theme: understanding the effect of change on a given development artifact, conceive a way to trace such change, and develop techniques to incrementally deal with its impact. Evolution is categorized as being driven by (i) a change in the context (environment), (ii) a change in the behavioral or security requirements, and (iii) a change in the specification (design and implementation) of the system.

In order to achieve the ambitious objective stated in the previous section and in light of the above-mentioned flavors of change, the project has defined the following array of integrated activities, as summarized in Figure 1, which visualizes how these activities fit together in the life-cycle of an adaptive security-critical system.

Industrial Scenarios Validation (WP1). The techniques and tools developed within the project are validated in the context of three case studies: (i) evolvability of software running on portable devices like smart cards, (ii) evolvability of software running on a residential gateway providing digital home services, and (iii) evolvability of software running on the workstation of air traffic control operators.

Architecture and Process (WP2). As configuration management can no longer be separated from the development of the application, one objective is a configuration management process that is tightly intertwined with the development and life-cycle processes defined in the project. This process is supported by model-based tools that allow for automated reconfiguration as a reaction to change as well as verification of security properties in a changing context. To successfully validate this process in the proposed case-studies, it needs to be instantiated on the basis of a common architecture, which is rich enough to address the challenges of incompatible versions, hardware reconfiguration, scalability, and heterogeneity.

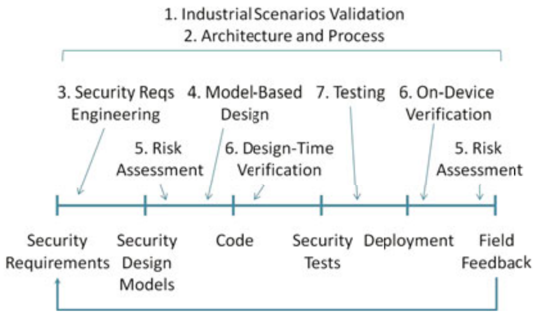

Fig. 1. Structure of the SecureChange project

Security Requirements Engineering (WP3). The project defines a new method that addresses the two possible interpretations of the concept of "evolving requirements". The first are the inventions that are necessary to consider in the current design, which are unknown but potentially vital future requirements. The second is how to redefine the flow from requirements to system, so that we can reconstruct how the system can evolve when requirements evolve.

Model-Based Design (WP4). The project develops a model-based design approach that is tailored to the needs of the secure development of adaptable systems. The approach glues requirement techniques to the final code and provides tools that allow the user to automatically analyze the models against its a-priori requirements, also taking into account the change scenarios that the risk analysis has defined likely. In particular, the approach takes into account various system views, including infrastructure elements, stakeholders, and hardware configurations—all of which may be subject to change during the system evolution.

Risk Assessment (WP5). The project also has the goal of generalizing existing methods and techniques for security assessment to cope with longevity. This addresses the issue of system documentation to facilitate assessment of security modulo changes as well as techniques, methods and tools for automatic or semi-automatic revalidation of existing assessment results modulo changes.

Design-Time and On-Device Verification (WP6). The project attempts to identify software programming models that can provably resist future errors, and how we can change the V&V process in order to cope with changes. Furthermore, the project provides embedded verification techniques targeting two goals. First, new code on an autonomous system must be verified in order to be safely loaded, and second, when new security requirements are provided, the verification process itself must be adapted.

Testing (WP7). The project also provides solutions for evaluating the impacts of changes, by means of model-based testing. By analyzing the differences between each model and their evolutions, testing strategies are defined that highlight the evolution or non-regression of the system.

3 Key Results

During the first project year, the SecureChange project obtained key results within all work packages. In the rest of this Section, a few highlights are presented:

1. Industrial Scenarios Validation. Based on change scenarios observed in the context of the industrial case studies, a taxonomy of change has been defined as a consortium-wide effort. The taxonomy provides the "scope" of the project activities. The classification has two sides: (i) how things change (problems) and (ii) how we deal with changes (solutions).
2. Architecture and Process. A methodology based on so-called Change Patterns has been defined to support adaptability to changing requirements at

the architectural level (i.e., co-evolution of security at the requirements and architectural level). Further, the framework for an artefact-centric change management process has been defined. The process keeps track of, and orchestrates the ripple effects among artifacts when a change is injected at any level.
3. Security Requirements Engineering. An approach to manage the evolution process of the requirements specification has been defined. The approach leverages so-called Change Rules. Evolution rules are specified as event-condition-action tuples. Events are changes in the requirements artifact (e.g., something is added and consequently a security inconsistency is detected) that trigger corrective actions, (e.g., the artefact is transformed to preserve the key security requirements).
4. Model-Based Design. Based on UMLsec, the UMLseCh extension has been defined. The extended notation allows to insert expected changes as predefined markers in the design models. These markers are leveraged for the formal security analysis carried out by a companion tool. The markers provide an indication of model elements that are added, deleted, or replaced. The tool performs an incremental analysis starting from both the original model (assumed correct) and the model of change.
5. Risk Assessment. A systematic approach has been defined (among others, as an extension of the CORAS method for security risk analysis) to identify the updates made to the target of evaluation, identify which security risks and parts of the security risk model are affected by the updates, and update the security risk model without having to change the unaffected parts.
6. Design-Time and On-Device Verification. At design-time, a programming model (failboxes) for verifying absence of dependency safety violations has been defined. At run-time, a programming model for writing safe programs that dynamically (un)load modules has been defined. Note that dynamically (un)loading modules is a key feature of adaptable systems.
7. Testing. An algorithm has been defined that, given the old test suite and the new design model, is able to compute: security tests that are outdated in light of the modeled change (obsolete), tests that are unchanged (reusable), tests that must be adapted or added (evolution).

The above results have led to several scientific publications that are available through the project portal[1].

[1] `http://www.securechange.eu`