

# Introduction to Software Engineering for Secure Systems

## SESS06 – Secure by design

Danilo Bruschi  
Dip. Informatica e  
Comunicazione  
Università degli Studi di Milano  
Via Comelico 39/41 – I-20135  
Milan, Italy  
bruschi@dico.unimi.it

Bart De Win  
Katholieke Universiteit Leuven  
Celestijnenlaan 200A –  
B-3001  
Leuven, Belgium  
bart.dewin@cs.kuleuven.be

Mattia Monga  
Dip. Informatica e  
Comunicazione  
Università degli Studi di Milano  
Via Comelico 39/41 – I-20135  
Milan, Italy  
monga@dico.unimi.it

### Categories and Subject Descriptors

D.2.m [Software Engineering]: Miscellaneous

### General Terms

Security, Design

### Keywords

Security requirements, trustworthiness, secure programming, security testing, security usability

## 1. OUTLINE OF THE THEME AND GOALS

The issue of software security is increasingly relevant in a world where most of our life depends directly on several complex computer-based systems. Today the Internet connects and enables a growing list of critical activities from which people expect services and revenues. In other words, they *trust* these systems to be able to provide data and elaborations with a degree of confidentiality, integrity, and availability compatible with their needs.

Historically, the software engineering community has strived more to obtain validity than trustworthiness. Nowadays, however, software ubiquity in the creation of critical infrastructures and ease of third-party service integration has raised the value of trustworthiness and new efforts should be dedicated to achieve it. In particular, security concerns should be taken into account as early as possible, and not added to systems as an after-thought: this is extremely expensive and it may compromise the design integrity in critical ways. Moreover, security features such as cryptographic protocols and tamper-resistant hardware cannot be simply used to “decorate” applications, to transform an insecure product in a secure one just by this addition. Surprisingly enough, several security holes are recurrent, notwithstanding the experience accumulated by security research in the

last decades. Software engineers and practitioners should assimilate basic security techniques and integrate them in the current practice, while understanding associated costs and benefits.

Conversely, computer security still lacks of practical methodological approaches which could help in reducing security exposures in many critical applications. Even well defined formalisms such as the security standards (e.g., Common Criteria [2] and BS7799 [1]), are challenging to integrate with mainstream software engineering practices. In such a context, several well-known software engineering disciplines such as verification, testing, program analysis, process support, configuration management, requirement engineering, etc. could contribute in improving security solutions.

The SESS workshop aims at providing a venue for software engineers and security researchers to exchange ideas and techniques. The first edition was held in conjunction with ICSE2005 [3]. The workshop website is <http://homes.dico.unimi.it/~monga/sess06.html>.

This year we received 16 submissions from 42 authors, coming from 10 different countries all around the world (Europe, North and South America, and Asia). The program committee selected 9 papers which are going to be published in these proceedings and will be discussed during the workshop. The contributed articles address the problems summarized in the following.

### 1.1 Security requirements engineering

Surprisingly enough, people often talk about security requirements, but there is no consensus of what they are and how one can derive them. Haley et al. [7] provide a rich and thorough discussion of what can be considered a “security requirements”. They also present a framework for security requirements elicitation and analysis in which security are expressed in terms of constraints about system goals.

A non trivial problem in the real world, addressed by Lee et al. [8], is extraction and deep understanding of security requirements from regulatory documents. The proposed approach combines several techniques, including ontological tools. The paper suggests a methodology for providing effective ways to understand security requirements, gather relevant evidences, perceive related risks in the operational environment, and reveal their causal relationships with other domain concepts as required by Certification and Accreditation (C&A) activities. The paper demonstrates the ap-

propriateness of the approach by building problem domain ontology from regulatory documents enforced by the Department of Defense Information Technology Security Certification and Accreditation Process (DITSCAP).

On almost opposite view is taken by Boström et al. [5], who propose a way of extending eXtreme Programming (XP) practices, in particular the original planning game and the coding guidelines, to aid the developers and the customer to engineer security requirements while maintaining the iterative and rapid feedback-driven nature of XP, best suited for small projects.

## 1.2 Lifecycle of secure software products

Since security is often an afterthought when developing software, and is often bolted on late in development or even during deployment or maintenance, through activities such as penetration testing, add-on security software and penetrate-and-patch maintenance, Ardi et al. [4] suggest an approach to focus on security from the beginning, by understanding on what might cause vulnerabilities, and how specific activities combine to prevent them. Vulnerability cause graphs, which encode information about vulnerability causes, and security activity graphs, which encode information about security activities can be used at this aim.

Software engineering today relies to a large extent on acquiring and composing software components and other software-related artifacts from different producers, either at design or at run time. Since one has to get some assurance that the third-party code is not harmful, Naedele and Koch [9] describe different trust models (what motivates suppliers) and approaches adopted to tamper-proof the delivery chain (who are we trusting). Such secure channels can theoretically be realized using digital signature technology, but some practical and theoretical challenges remain.

## 1.3 Modelling and analyzing security properties

Security requirements are often changed after the system deployment. Thus, the security enforcement mechanism is sometimes configured separately, as is the case of Java security policies. However, it is not always clear how a change in the security requirements impacts the overall security of the whole system. Nakajima and Tamai [10] propose to use Alloy for formalizing and checking the interaction between policies and their applicative enforcement.

Network Intrusion Detection Systems (NIDSs) can be composed of a potentially large number of probes, which monitor the traffic flowing in the network. Deciding where probes should be placed and what information they need in order to detect the desired attacks can be a demanding task for network administrators. Rolando et al. [11] present a logical framework that can be used to reason about the possible occurrence of (topology-dependent) attacks and the protection against them.

## 1.4 Aspect oriented programming and security

Aspect oriented programming (AOP) can be a powerful tool for building secure applications since security is often recognized as a cross-cutting concern. However, AOP is sometimes seen as a silver bullet that can be used to simply add security to an existing application. Checking for the opposite, that is, for the dangers of introducing additional

security flaws through AOP, however, is also necessary. De Win et al. [12] show several concrete examples of using privileged aspects and aspects that intercept security-relevant functions and propose an outlook to possible solutions to the problem.

Instead, Chen and Chen [6] use AspectJ to overcome advanced code obfuscation and attacking Java software effectively using its bytecode instrumentation mechanism. Simplicity and the very low cost of such AspectJ attacks make them worth wider attention from the Java and AspectJ community. The paper sketches also possible countermeasures.

## Acknowledgments

The organizers want to thank all the reviewers and the authors for their contribution to a workshop that promises to be very interesting for both the security and the software engineering research community.

## 2. REFERENCES

- [1] The BS7799 / BS 7799 security standard. <http://www.thewindow.to/bs7799/>.
- [2] The Common Criteria portal. <http://www.commoncriteriaportal.org/>.
- [3] SESS'05: Proceedings of the 2005 workshop on software engineering for secure systems: building trustworthy applications, 2005. Available at <http://www.acm.org/dl>.
- [4] Shanai Ardi, David Byers, and Nahid Shahmehri. Towards a structured unified process for software security. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [5] Gustav Boström, Jaana Wäyrynen, Konstantin Beznosov, Marine Bodén, and Philippe Kruchten. Extending xp practices to support security requirements engineering. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [6] Kung Chen and Ju-Bing Chen. On instrumenting obfuscated java bytecode with aspects. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [7] Charles B. Haley, Jonathan D. Moffett, Robin Laney, and Bashar Nuseibeh. A framework for security requirements engineering. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [8] Seok-Won Lee, Robin Gandhi, Divya Muthurajan, Deepak Yavagal, and Gail-Joon Ahn. Building problem domain ontology from security requirements in regulatory documents. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [9] Martin Naedele and Thomas Koch. Trust and tamper-proof software delivery. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [10] Shin Nakajima and Tetsuo Tamai. Formal specification and analysis of JAAS framework. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [11] Marco Rolando, Matteo Rossi, Niccolò Sanarico, and Dino Mandrioli. A formal approach to sensor placement and configuration in a network intrusion detection system. In *SESS'06*, Shanghai, China, May 2006. ACM.
- [12] Bart De Win, Frank Piessens, and Wouter Joosen. How secure is AOP and what can we do about it. In *SESS'06*, Shanghai, China, May 2006. ACM.