Software Engineering for Secure Systems

SESS05 – Building Trustworthy Applications

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.ac.be

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

1. OUTLINE OF THE THEME AND GOALS

Every software application is built and deployed to accomplish some goal pursued by its interested parties. Thus, software engineers aim at designing, implementing, and maintaining *valid* applications that meet the needs of stakeholders. However, every application can be also potentially *misused,* that is, used to pursue goals that contrast the ones intended by stakeholders. Therefore, software engineers should try to design applications that, while still valid, are also *trustworthy* and cannot be misused. Validity and trustworthiness are goals that often cannot be achieved either because they are too costly or because they stem from conflicting needs. Historically, the software engineering community has strived more to obtain validity than trustiness. Nowadays, however, software ubiquity in the creation of critical infrastructures has risen the value of trustworthiness and new efforts should be dedicated to achieve it.

The major source of vulnerability of systems has been recognized to be poor-quality software [1]. However, while secure applications are also valid and robust ones, security is a specific *non-functional* requirement that has to be explicitly and carefully taken into account during analysis, implementation, testing, and deployment [2]. Moreover, some of the most successful techniques used by software engineers may conflict with security objectives. *Abstraction*, for example, is the invaluable device the designers use in order to cope with complexity, but, since it is rarely applied as a pure mathematical generalization, it could force one to neglect details that can be exploited to misuse an application; *late binding*, while a fundamental tool in pursuing design for change, could be hijacked to adapt systems to malicious goals; *COTS*, commercial off-the-shelf components, if they might foster the profitableness of software industry, they also introduce black-box subsystems that are difficult to manage when reasoning about the chain of trust of the whole system.

The security research community has proposed a number of techniques, such as cryptography protocols and tamper-resistant hardware, that could be used to build trust in software components, tools and process. However, this knowledge cannot be simply used to *augment* software engineers' toolboxes, since applications "decorated" by applying security features could only create a false sense of trustworthiness. Instead, most of the approaches should be re-thought under the light shed by the experience of security researchers in order to empower practitioners with novel techniques that are able to tackle the problem of building valid and trustworthy systems, while understanding associated costs and benefits. At the same time, several well-known software engineering disciplines such as verification, testing, program analysis, process support, configuration management, requirement engineering, etc. could contribute to improving security solutions that sometimes lack a coherent methodological approach or, as in the case of security standards proposed by the Common Criteria [3] or BS7799 [4], are challenging to be integrated with mainstream software engineering practice.

This workshop will provide a venue to discuss techniques that enable the building and validation of secure applications by putting together people from the software engineering and the security fields fostering a fruitful cross-fertilization between the two communities.

2. REFERENCES

[1] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proceedings of the 26th International Conference on Software Engineering.* IEEE Computer Society, 2004, pp. 148–157.

[2] P. T. Devanbu and S. Stubblebine, "Software engineering for security: a roadmap," in *Proceedings of the conference on The future of Software engineering.* ACM Press, 2000, pp. 227–239.

[3] "The Common Criteria portal," http://www.commoncriteriaportal.org/.

[4] "The BS7799 / BS 7799 security standard," http://www.thewindow.to/bs7799/.

[5] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed., ser. Best Practices. Microsoft Press, 2003.