In Proceedings of the 29th International Conference on Software Engineering (ICSE 07)
Companion Volume, pp. 111-112, Minneapolis/St. Paul, MN, USA. IEEE Press. May 20-26, 2007.

The 3rd International Workshop on Software Engineering for Secure Systems SESS07 – Dependable and secure

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

Seok-Won Lee
Dept. of Software and Information Systems
University of North Carolina at Charlotte
9201 University City Blvd.
Charlotte, NC 28223, USA
seoklee@uncc.edu

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

The theme of this year of the International Conference on Software Engineering is about "Developing Dependable Software", acknowledging the fact that our lives depend directly on several complex software-based systems. The Internet connects and enables a growing list of critical activities from which people expect services and revenues. They should be able to *trust* these systems to provide data and elaborations with a degree of confidentiality, integrity, and availability compatible with their needs. The pervasiveness of software products in the creation of critical infrastructures has raised the value of trustworthiness and new efforts should be dedicated to achieve it. Yet, nowadays almost every application has some kind of security requirement even if its use is not to be considered critical.

Thus, designers have to cope with the complexity of insecure operating environments by considering threats to their application correctness. 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. Security features such as cryptographic protocols and tamper-resistant hardware cannot be simply added on to transform an insecure product to a secure one.

Security solutions and patterns are hard to reuse in different contexts, they crosscut all the system components and a single vulnerability might compromise the trustworthiness of the entire system. Thus, not surprisingly, 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 discover new techniques for integrating them in the current practice, while understanding associated costs and benefits.

The SESS workshop aims at providing a venue for software engineers and security researchers to exchange ideas and techniques. The past editions were held in conjunction with ICSE2005 and ICSE2006. The workshop website is `http://homes.dico.unimi.it/~monga/sess07.html`.

2 Summary of the contributed papers

This year we received 17 submissions from 50 authors, coming from 8 different countries all around the world. The program committee selected 10 papers which are going to be published in these proceedings. Seven [1, 2, 3, 6, 7, 9, 10] will be discussed during the workshop and three of them [4, 5, 8] were selected for a short presentation. The contributed articles address the problems summarized in the following.

Security is hard to be encapsulated in a reusable component: Raffetseder et al.[7] describe the challenges the they faced in porting from the Mozilla browser to MS Internet Explorer an anti-phishing plugin.

Vulnerabilities are critical weaknesses in applications and specific testing should be performed to discover them

prior to deployment. Lanzi et al. [3] propose to use a static and dynamic analysis to perform *smart fuzzing,* in order to identify security relevant flaws in binary programs. Martin and Xie [6] suggest to use change-impact analysis for automatic generation of test cases for access control policies.

Security problems in Supervisory Control and Data Acquisition (SCADA) of critical infrastructures could be catastrophic. Xiao et al. [10] advance the idea that the survivability of the system can be enhanced if a non-intrusive workflow layer is used for predicting failures via simulation of attack scenarios.

Monitoring is crucial to security enforcement and Vanoverberghe, Piessens [9] propose a new language element, the check block, that developers can use to make their applications monitor aware.

Traditional software development processes have little support for meeting security requirements. Gregoire et al. [1] survey two process frameworks for secure software engineering, namely CLASP and SDL, and outline suggestions for improvement.

Heyman et al. [2] examine published security patterns and discuss why their adoption does not live up to their potential.

Finally, Thomas and Williams in [8] discuss how to automatically fix dangerous SQL statements, Wang and Li propose a threat model driven approach to security testing in [5], and Lee et al. in [4] describe their requirements-driven workbench for supporting the software certification and accreditation process required by the US Dep. of Defense.

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.

References

[1] Johan Gregoire, Koen Buyens, Bart De Win, Riccardo Scandariato, and Wouter Joosen. On the secure software engineering process: CLASP and SDL compared. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[2] Thomas Heyman, Koen Yskout, Riccardo Scandariato, and Wouter Joosen. An analysis of the security patterns landscape. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[3] Andrea Lanzi, Lorenzo Martignoni, Mattia Monga, and Roberto Paleari. A smart fuzzer for x86 executables. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[4] Seok-Won Lee, Robin Gandhi, and Siddharth Wagle. Towards a requirements-driven workbench for supporting software certification and accreditation. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[5] Xuandong Li Linzhang Wang. A threat model driven approach for security testing. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[6] Evan Martin and Tao Xie. Automated test generation for access control policies via change-impact analysis. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[7] Thomas Raffetseder, Engin Kirda, and Christopher Kruegel. Building anti-phishing browser plug-ins: An experience report. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[8] Stephen Thomas and Laurie Williams. Using automated fix generation to secure sql statements. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[9] Dries Vanoverberghe and Frank Piessens. Supporting security monitor-aware development. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.

[10] Kun Xiao, Nianen Chen, Shangping Ren, Limin Shen, Xianhe Sun, Kevin Kwiat, and Michael Macalik. A workflow-based non-intrusive approach for enhancing the survivability of critical infrastructures in cyber environment. In *SESS'07: Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, Minneapolis, MN, USA, May 2007.