Understanding Software Security from Design to Deployment

Mehdi Mirakhorli
Rochester Institute of Technology
Rochester, NY, USA
mehdi@se.rit.edu

Matthias Galster
University of Canterbury
Christchurch, New Zealand
mgalster@ieee.org

Laurie Williams
North Carolina State University
Raleigh, NC, USA
lawilli31@ncsu.edu

DOI: 3385678.3385687
http://doi.org/3385678.3385687

ABSTRACT

Analyzing, implementing and maintaining security requirements of software-intensive systems and achieving truly secure software requires planning for security from ground up, and continuously assuring that security is maintained across the software's lifecycle and even after deployment when software evolves. Given the increasing complexity of software systems, new application domains, dynamic and often critical operating conditions, the distributed nature of many software systems, and fast moving markets which put pressure on software vendors, building secure systems from ground up becomes even more challenging. Security-related issues have previously been targeted in software engineering sub-communities and venues. In the second edition of the International Workshop on Security from Design to Deployment (SEAD) at the International Conference on Automated Software Engineering (ASE) 2020, we aimed to bring the research and practitioner communities of requirements engineers, security experts, architects, developers, and testers together to identify foundations, and challenges, and to formulate solutions related to automating the analysis, design, implementation, testing, and maintenance of secure software systems.

Categories and Subject Descriptors

• Software and its engineering~Software organization and properties • Software and its engineering~Extra-functional properties • Software and its engineering~Software creation and management • Software and its engineering~Designing software

General Terms

Management, Measurement, Performance, Design, Security, Verification.

Keywords

Software security, requirements, architecture, design, deployment, workshop, ASE.

1. INTRODUCTION

Security has previously been targeted by various software engineering sub-communities (e.g., security requirements engineering, software architecture), as well as different community events, e.g., conferences related to cybersecurity, networking and internet technologies (e.g., the International Symposium on Software Reliability Engineering [ISSRE], the International Conference on Dependable Systems and Networks [DSN]). Each of these communities have their own paradigms (meta-models, languages, methods). Architecture conferences, e.g., the International Conference on Software Architecture (ICSA) focuses primarily on high-level concepts to implement security (e.g., design or architecture patterns), while conferences such as the International Conference on Software Testing (ICST) and the International Conference on Software Maintenance and Evolution (ICSME) focus on more specific low-level programming issues.

Many developers are not familiar with secure coding practices and do not understand their application's security architecture [1]. This knowledge gap results in security issues such as an implementation that deviates from the initial security architecture design, missing security choices in the code (e.g., architecture tactics) or incorrectly implementing security countermeasures [2].

The first edition of the International Workshop on Security from Design to Deployment (SEAD) was co-located with the International Conference on Software Engineering (ICSE) 2018 in Gothenburg, Sweden. In this second edition of SEAD held in conjunction with the International Conference on Automated Software Engineering (ASE) in San Diego, CA, we aimed to bring the research and practitioner communities of requirements engineers**,** security experts**,** architects**,** developers, and testers together to identify foundations, challenges and formulate solutions related to automating the analysis, design, implementation, testing, and maintenance of secure software systems. The workshop website is available online: https://2019.ase-conferences.org/home/sead-2019.

2. BACKGROUND

Secure design refers to a design that supports and "enforces the necessary authentication, authorization, confidentiality, data integrity, accountability, availability, and non-repudiation requirements, even when the system is under attack" (IEEE Center for Secure Design).[1] Security-by-design requires that security awareness is injected into software development from early inception phases. Also, it is important to ensure that all stakeholders in all phases of the life cycle of a software product or service are aware of security requirements, the current security architecture, threats associated with the system, potential vulnerabilities and mitigation techniques to address them. Addressing the current and future needs of is non-trivial. This requires addressing challenging scientific and engineering problems, but also vulnerabilities that arise from human behaviors and choices.

There has traditionally been a distinction between high-level architecting and low-level implementation and maintenance activities such as coding to ensure that security is not put at risk through vulnerable implementations in code. In fact, software design, implementation and testing are established areas in software engineering research, education and practice. However, problems tend to occur when there is a mismatch between decisions made in each of these areas, or inappropriate techniques and tools are used to carry out, realize and test the decisions from one area to another one. Those who are developing and maintaining software are often not engaged in early security design and planning phases of the software. In particular, junior software developers tend to lack "security thinking" and awareness and software assurance skills.

3. WORKSHOP GOALS

The goals of SEAD 2019 were:

- Provide an open forum for discussions: We included discussions around software security, including automated and lean/agile practices to address security; continuous development/deployment and DevOps; impact of technology advances on achieving security; automated validation and verification in the context of security.
- Share knowledge and experiences: The workshop offered an opportunity to become familiar with how security affects development processes and practices, what current challenges

[1]https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf

are and how to address these challenges. In particular, the workshop discussed how to bring "security thinking" into daily development activities. We discussed problems, solutions and share lessons learned.

- Grow community: This is the second edition of SEAD, we grew a cross-cutting multi-disciplinary community. We discussed secure software in a broader context of software engineering and across application and technology domains to leverage experiences made in different communities.

- Build a corpus of artefacts: Since security is defined quite broadly, we not only aim at sharpened our understanding of what types of systems are there and how to build and maintain them, but also started building a corpus of artifacts that can support the automated design, implementation and maintenance of such systems (see below). This corpus can include architectures, designs, code, etc.

4. KEYNOTE

The workshop started with a keynote by Hamid Bagheri from the University of Nebraska-Lincoln, USA on Automating Pragmatic Software Dependability. Hamid argued that the inherent complexity of large-scale software systems has always posed a significant challenge to software practitioners. On top of this, the ever increasing expansion of software into nearly every aspect of modern life is making its dependability more critical than ever. Automating the cumbersome and error-prone software engineering activities is paramount for achieving dependable software. Lightweight formal methods which are automated, yet provide formally-precise analysis capabilities, have recently received a lot of attention in the software engineering community. Hamid presented ongoing research which explores the possibility of leveraging such formally rigorous techniques that rely on recent advancements in constraint solving technologies for automated and practical dependability analysis of widely-used software systems.

5. DISCUSSIONS

After the keynote and six paper presentations, participants discussed various topics related to security from design to deployment:

- Security metrics and how to measure improvements in the security of software systems
- Security data, including access to security data and how to collect high-quality security data
- Insider threat detection (i.e., the human aspects related to the intentional and unintentional behavior of software developers when implementing security)
- Hardware-vulnerability-aware software (i.e., mitigating hardware vulnerability threats in software)
- Security versus other quality attributes (e.g., usability) and trade-offs between security and other quality attributes
- Cost-effective secure software development and the economics of secure software
- Tools and techniques for "Security by Design"
- Security in specific domains (e.g., threat modelling, benchmarking in cyber-physical systems [CPS], defense mechanisms in CPS, incidence response in CPS, fuzz testing to detect vulnerabilities and design flaws [e.g., in Android, blockchain/smart contracts])
- Composability of security analysis (e.g., plug-in architecture, systems-of-systems, ecosystems) and how security in different components impacts system security
- Security mining (for early security awareness and for the comparison of different software versions regarding security)

These topics could provide the starting point for defining an agenda or roadmap for future work on ensuring security from design to deployment.

Furthermore, participants started to create a "Call for Collaborators" for potential work related to the workshop. The "call" included topics, problems, research questions and requests for help with solving security-related research problems (e.g., data analysis, labeling, coding). Some examples are provided below:

- Effective automatic labeling of high quality data
- Vulnerability descriptions and representing vulnerabilities in source code (for machine learning)
- Automating attack and defense synthesis techniques in cyber-physical systems
- Incident response and recovery in cyber-physical systems
- Improving conventional Human-Machine Interfaces (HMI) with security mechanisms; human factors and usability in cyber-physical systems

Finally, participants started to create a list of publicly available data and tools that might be useful for other researchers and practitioners:

- Vulnerability Detection Dataset (VDD), labeled for the five most common CWEs, including more than 1,3 million code snippets/examples at method level: https://osf.io/d45bw/
- Dataset to support research in the design of secure CPS: https://itrust.sutd.edu.sg/itrust-labs_datasets/

The above lists are available online in a shared spreadsheet: https://docs.google.com/spreadsheets/d/1rk2HibAWNGeFgLayEExiOrN3CNBdDqCosAkh2GhUx1o/edit?usp=sharing

6. ACKNOWLEDGMENTS

We thank workshop participants and all authors for submitting their work to SEAD 2019. We also thank the members of the international program committee and the ASE workshop chairs and organizers for supporting our workshop.

7. REFERENCES

[1] J. Santos et al., "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird", IEEE International Conference on Software Architecture (ICSA) 2017.

[2] J. Santos et al., "A Catalog of Security Architecture Weaknesses", IEEE International Conference on Software Architecture Workshops (ICSAW) 2017.