Software Security Requirements Engineering: State of the Art

Muthu Ramachandran

School of Computing, Creative Technologies and Engineering
Leeds Beckett University
Leeds LS6 3QS UK
Email: M.Ramachandran@leedsbeckett.ac.uk

Abstract. Software Engineering has established techniques, methods and technology over two decades. However, due to the lack of understanding of software security vulnerabilities, we have not been so successful in applying software engineering principles that have been established for the past at least 25 years, when developing secure software systems. Therefore, software security can not be just added after a system has been built and delivered to customers as seen in today's software applications. This keynote paper provides concise methods, techniques, and best practice requirements guidelines on software security and also discusses an Integrated-Secure SDLC model (IS-SDLC), which will benefit practitioners, researchers, learners, and educators.

Keywords : Software Security Engineering, Software security requirements engineering, Secured Software Development, SQUARE method, BSI, Touchpoint, SDL

1 Introduction

There is no doubt that the internet technology has revolutionised human lives, communications, digital economy, socialisation, and entertainment. At the same time demands for internet enabled applications grows rapidly. Almost all businesses, applications, entertainment devices, mobile devices, robots, large scale systems (aircrafts, mission control systems), safety-critical systems, medical systems, internet of things devices are internet enabled for various reasons such as online upgrade, distributed applications, team projects, and server connectivity. Therefore, there is ever growing demand for secured applications and trust. Cyber attacks are increasing continuously from spam, phishing, identify theft, and others in much larger scale attacks such as money laundering and cyber terrorism. Foritfy report (2009) says that there is a real possibility that a cyber attack could disable command systems, bring down power grids, open dam floodgates, paralyse communications and transport systems, creating mass hysteria: Any or all of which could be the precursor to terrorist or mili-

tary attack. These are some of the threats since we (personal, govt. organisations, companies, and business) mostly depend on computers and mobiles for communications and management.

This keynote paper aims to outline the importance of developing secured software systems using a disciplined approach known as software security engineering and it is also known as secure software development. In particular, this paper identifies key methods and techniques on software security requirements engineering as it is the heart of developing secure software systems. This paper discusses clear best practice guidelines on software security and discusses our Integrated-Secure SDLC (IS-SDLC) model which overcomes current difficulties in identifying and visually representing security process which have been elaborated from security requirements.

2 Why Software Security Engineering?

Software Engineering has established techniques, methods, and technology over two decades. However, security issues are direct attributes of various software such as applications, user interface, networking, distribution, data-intensive transactions, and communication tools, etc. Current applications are being developed and delivered where security has been patched as aftermath. Early commercial developers have tackled security problems using firewalls (at the application level), penetration testing, and patch management.

We are also faced with tackling fast growing information warfare, cybercrime, cyber-terrorism, identify theft, spam, and other various threats. Therefore, it is important to understand the security concerns starting from requirements, design, and testing to help us Build-In Security (BSI) instead of batching security afterwards. McGraw [1] says *a central and critical aspect of the computer security problem is a software problem.* This paper defines *software security engineering as a discipline which considers capturing and modelling for security, design for security, adopting best practices, testing for security, managing, and educating software security to all stakeholders.*

Software engineering has well established framework of methods, techniques, rich processes that can address small to very large scale products and organisations (CMM, CMMi, SPICE, etc), and technology [modelling (UML), CASE tools, and CAST tools, and others]. Software Engineering has also been well established quality models and methods, reuse models and methods, reliability models and methods, and numerous lists of other techniques. The so called -lities of software engineering long has been contributed as part of quality attributes (Quality, Testability, Maintainability, Security, Reliability, Reusability). These attributes can't be just added on to the system as they have to be built in from early part of the life cycle stages (a typical software development lifecycle include starting from requirements engineering (RE), software specification, software & architectural design, software development (coding), software testing, and maintenance. Security has become highly important attribute since the development of online based applications. Software project management

has well established techniques and breadth of knowledge including global development (due to emergence of internet revolution and people skills across the globe), cost reduction techniques, risk management techniques, and others. Nowadays, most of the current systems and devices are web enabled and hence security needs to be achieved right from beginning: need to be identified, captured, designed, developed and tested. Ashford [2] reports UK business spends 75% of the software development budget on fixing security flaws after delivering the product. This is a huge expenditure and it also creates untrustworthiness amongst customers.

Allen et al. [3] state that the one of the main goals of Software Security Engineering is to address software security best practices, process, techniques, and tools in every phases and activities of any standard software development life cycle (SDLC). The main goal of building secured software which is defect free and better built with:

- Continue to operate normally in any event of attacks and to tolerate any failure
- Limiting damages emerging as an outcome of any attacks triggered
- Build Trust & Resiliency In (BTRI)
- Data and asset protection

In other words, secured software should operate normally in the event of any attacks. In addition, it involves the process of extracting security requirements from overall system requirements (includes hardware, software, business, marketing, and environmental requirements) and then also further refined and extracted security and software security requirements from software and business requirements. Then the refined software security requirements can be embedded and traced across the software development life cycle (SDLC) phases such as requirements, design, development, and testing. This has not explained well in security related literatures so far. This provides a clear definition of eliciting software security requirements.

3 Software Security Requirements Engineering

Requirements are the starting point, responsible for any system, legal and contractual issues, governance, and provide full functional perspective of the system being developed. Requirements Engineering is a discipline in its own right, which provides process, tools, techniques, modelling, cost estimation, project planning, and contractual agreements. There are wealth of requirements engineering methods, techniques, best practice guidelines, and tools [4-7]. However, due to the nature of increased demands for security-driven applications, current techniques are inadequate for capturing security related requirements effectively. Firesmith [8 & 9] reports that poor requirements are the main reasons for cost and schedule overruns, poor functionality and delivered systems that are never used. Requirements are classified into two major parts such as functional requirements which deal with the functionality of the system and non-functional requirements which deal with constraints, quality, data, standards, regulations, interfaces, performance, reliability, and other implementation requirements. Studies [4-7] have shown that requirements engineering defects cost 10 to 200

times to correct the system after implementation. Therefore, it is paramount to get the requirements correct, concise, and unambiguous.

Capturing business security requirements is a collaborative effort involves many stakeholders such as business analysts, software requirements engineer, software architect, and test managers. Security requirements should provide a clear set of security specific needs and expected behavior of a system. The main aim is to protect systems assets (data and files) and unauthorised access to the system from intentional attacks to the application software systems and other forms of internet based security attacks such as spam, denial of service, identity theft, viruses, and many other forms of intentional attacks that emerges every day. Security remains a software problem as the number of threats and vulnerabilities reported to CERT (Computer Emergency Response team) [10 & 11] of 2493% increase between 1997 (311 cases reported), 2006 (8064 cases reported), and as of 30th April 2015 (3192 cases reported).

In traditional RE, security requirements are considered to be a part of non-function properties and are considered an aspect of implementation strategies such as password protection, authentication, firewalls, virus detection, denial-of-service attacks, etc. Therefore, security needs to be considered as highly specific set of requirements for every functional requirements that has been identified, and has to be applied throughout the life cycle so that we can achieve build in security (BSI). *In addition current RE methods have considered mostly what the system must do, but what the system must not do*. This is the key issue that will be considered when selecting RE methods for software security. Moreover, in Software security RE methods, there are more stakeholders than traditional RE methods have considered such as social engineers, security specialist, business process modeling experts, service computing specialists, and users. Often attackers look for defects in the system, not the system features and functionalities. A number of techniques have emerged to address RE from an attacker's perspective:

- Attack patterns are similar to design patterns which has been designed to study attacks from destructive mode, Allen et al. [3] and BSI [12].
- Misuse and abuse cases are a set of use cases from an attackers perspective, McGraw [1]
- Attack trees provide a formal mechanism for analysing and describing various ways in which attacks can happen from an attacker's perspective. Simply represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes, Schneier [13-14] and Ellison and Moore [15]
- Microsoft SDL provides support on threat modelling which describes a set of security aspects by defining a set of possible security attacks. This has an integral part of Microsoft's SDL method, Howard and LeBlanc [16]
- Building Security In (BSI) method [12], process, design principles, and techniques provided by McGraw [1] and others which is now officially supported

by the US department of Homeland security. Some of the design principles include:

- Correctness by Construction (CbyC)
- Securing the Weakest Link
- Defense in Depth
- Failing Securely
- Least Privilege
- Separation of Privilege
- Economy of Mechanism
- Least Common Mechanism
- Reluctance to Trust
- Never Assuming that your Secrets are Safe
- Complete Mediation
- Psychological Acceptability
- Promoting Privacy

- The SEI's (Software Engineering Institute) has identified a method known as SQUARE (Secure Quality Requirements Engineering) [17] which is to elicit and prioritise requirements and its consists of nine steps as follow:

  – Agree on definition
  – Identify security goals
  – Develop artefacts
  – Perform risk assessments
  – Select an elicitation technique
  – Elicit security requirements
  – Categorise security requirements
  – Prioritise security requirements
  – Inspect security requirements
  – Clear identification of requirements of the whole applications system and extract security requirements. Interact with stakeholders to clarify security requirements and the technology they want to use, and cost implications.

- OCTAVE method by Caralli at al. [18], Alberts and Dorofee [19] and Woody and Alberts [20] provides clear activities on security requirements:
  - Identify critical assets
  - Define security goals
  - Identify threats
  - Analyze risks
  - Define security requirements
- Other methods include CLASP [21] and S-SDLC [22] and have given detailed descriptions by Ramachandran [23]

Chen [24] distinguishes the key difference between software security engineering with that of robustness for software safety engineering. Software security engineering deals with engineering approach to software development with an aim to engineer and implement security features whereas robustness deals with engineering software for safety critical systems. Therefore, we need to identify, analyse, and incorporate secu-

rity requirements as part of the functional requirements process. Belapurkar et al. [25] have identified a list of some high-level areas for each security specific functional requirements as follows:

- Identification should address how a system recognises the actors/entities (humans or systems) interacting with the system.
- Authentication should address how a system validates the identity of entities
- Authorisation should address what privileges are to be set to an entity interacting with a system
- Non-repudiation should address how a system prevents entities from repudiating their interactions with the system functionality
- Integrity should address how a system protects information from any intentional or unintentional modifications and tampering
- Auditing should address how a system allows auditors to see the status of the security controls in place
- Privacy should address how a system prevents the unauthorised disclosure of sensitive information
- Availability should address how a system protects itself from intentional disruptions to service so that it is available to users when they need it.

Software security requirements are not only a set of constraints on the software systems but they satisfy required governance and provides protection and trust. This means that we need far more newer techniques such as attack patterns, misuse and abuse cases as part any requirements process.

4 Integrated Security Software Development Lifecycle Process

The above discussed drawbacks and requirements for a concise method, lead us to develop a model that integrates various activities of identifying and analysing software security engineering into software development process, and this new process and its activities is shown in Figure 1. However, this paper focuses on only software security requirements specific activities. According to this model, SSRE (software security requirements engineering) consists of identifying standards and strategies of the organisation with regards to requirements elicitation (including analysis, validation, verification), conducting risk management and mitigation, and identifying software security requirements consists of a further sub-processes of defining security, identifying security strategies, conducting areas and domain scope analysis, business process modeling and simulation, identifying security issues, applying use cases and misuse cases, attack patterns.

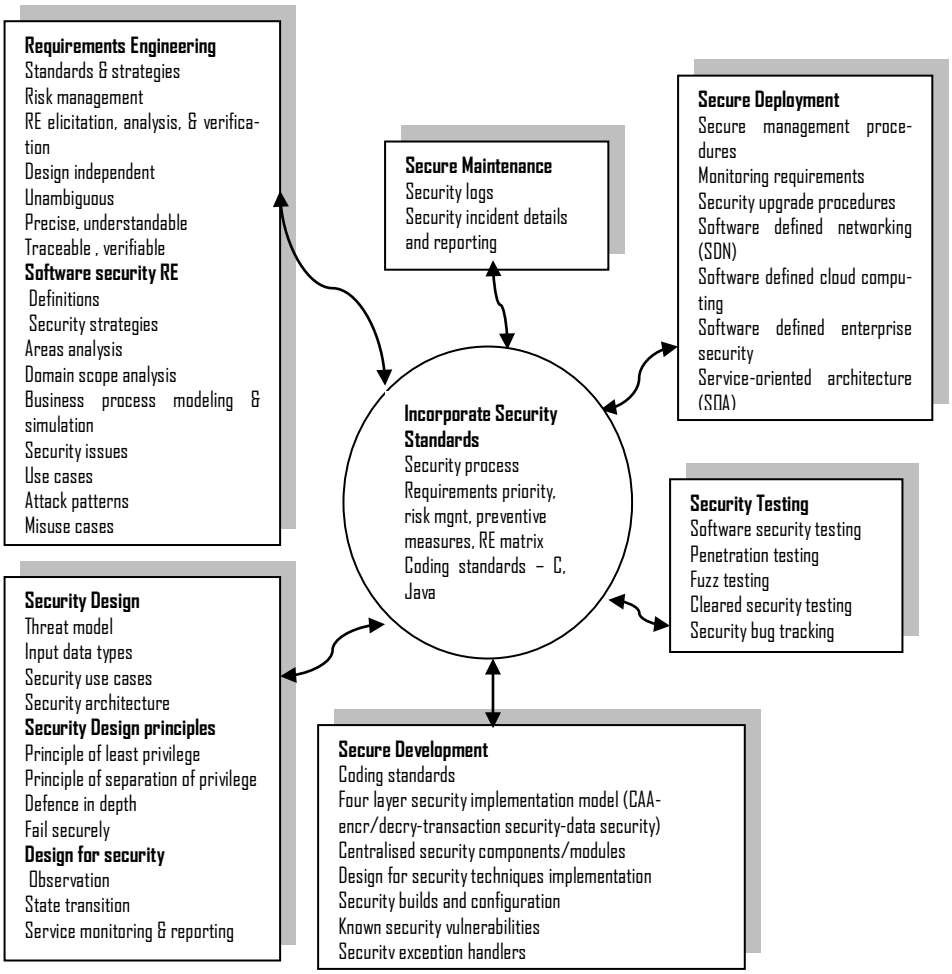

Fig. 1. Integrated secure software development engineering life cycle (IS-SDLC)

Likewise, this model also provides security-specific processes for identifying security threats during design, development, testing, deployment, and maintenance. There are a numerous number of good design principles that can be found in a vast majority of software design literatures. However, the following is a list of some of the key design principles that are highly relevant to software security design and are part of our IS-SDLC model:

- Principles of least privilege states to allow only a minimal set of rights (privileges) to a subject that requests access to a resource. This helps to avoid intentional or intentional damage that can be caused to a resource in case of an attack.
- Principles of separation of privilege states that a system should not allow access to resources based on a single condition rather it should be based on multiple conditions which has to be abstracted into independent components.

- Design by incorporating known CVE
- Design for resilience for which we have team up with IBM [26] to develop a resilience model which supports system sustainability along side with Building Trust and Security In (BTSI)
- Select software security requirements after performance simulation using BPMN (Business Process Modeling Notation) and is described in detail by Ramachandran [27]

SSRE activities in our IS-SDLC supports security in software defined networking (SDN), Cloud computing services (Software as a service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), Enterprise security includes cloud service providers and service consumers, and design for security principles and techniques. This the unique contribution of this model and for the body of knowledge in software security research.

5 Software Security Requirements Engineering Method as part of IS-SDLC

Software development and secure software development involve many stakeholders and business leaders and their coordination is critical for delivering secure software systems. The various stakeholders is shown in the Figure 2.

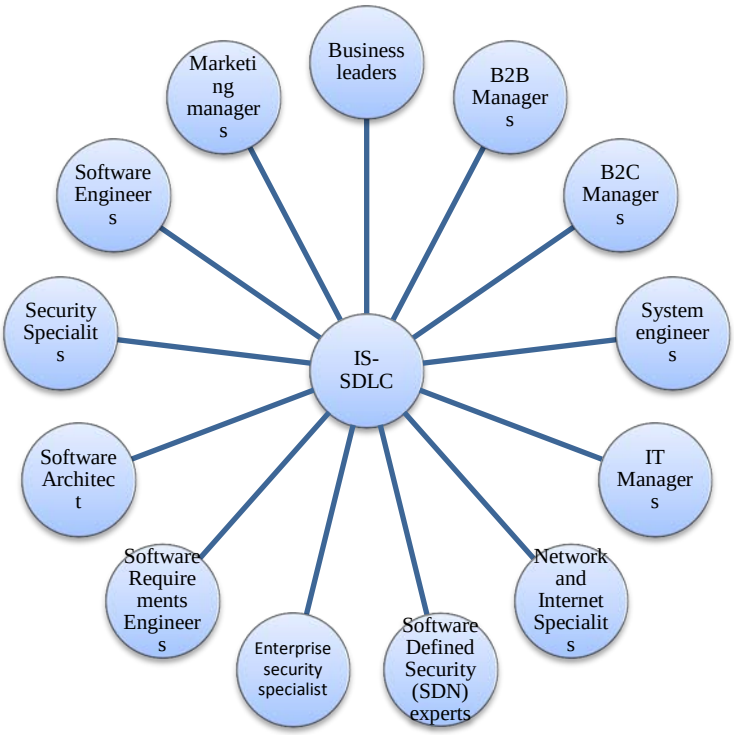

Fig. 2. Stakeholders in Integrated Secure-SDLC (IS-SDLC)

The previous section has provided a brief account of various methodologies [1-27] for eliciting requirements for software security. Most common best practices are:

1. Eliciting and extracting requirements for software security explicitly with visual notations
2. Prioritising software security requirements
3. Risk assessment and mitigation for software security requirements
4. Design and implement software security requirements
5. Providing SDLC life-cycle support

Existing methods lack heavily on incorporating social engineering to study software security requirements (learning from real experiences), security-specific business process modeling, performance simulations of the security-specific business processes, service computing, current and emerging technologies such as cloud computing, software-defined networking architecture, and software-defined enterprise security, and emerging vulnerabilities, and cyber attacks. This leads us to develop an integrated-secure software development model supporting software security requirements to be assessed and implemented explicitly in our method as presented in the Figure 1. Ramachandran [23] provides a comparative analysis of various SSRE meth-

ods based on our evaluation criteria used and this will help organization and engineers to choose appropriate method that is suitable for the system being developed. Based on the experience with IS-SDLC model in various projects, this paper has identified a set of best practice guidelines and recommendations on SSRE and SSE in general.

6 Best Practice Guidelines

For secured systems, this paper identifies a set of common guidelines that are applicable to most of the secure software development:

1. Develop a list of security requirements checklists and classify them as: critical, medium, and moderate.
2. Bring in requirements inspection team to conduct the security requirements validation process
3. Identify, elicit, analyse, and manage security requirements
4. Specify and model misuse cases and derive security requirements from misuse cases
5. Cross-check operational and functional requirements against security requirements
6. Establish an organisational security culture (e.g, check to make sure proper use of email systems do's and don'ts).
7. Apply business process Modelling and simulation using BPMN tools such as Bonita soft which provides clear performance attributed for all selected security-specific processes.

7 Conclusion

Software security engineering offers several best practices, techniques, and methods to develop systems and services that are built for security, resiliency, sustainability. However, software security can not be just added after a system has been built and delivered to customers as seen in today's software applications. This keynote paper provided concise techniques and best practice requirements guidelines on software security and also discussed an Integrated-Secure SDLC model (IS-SDLC), which will benefit practitioners, researchers, learners, and educators.

References

1. McGraw, G (2006) Software security: building security in, Addison Wesley, USA
2. Ashford, W (2009) http://www.computerweekly.com/Articles/2009/07/14/236875/on-demand-service-aims-to-cut-cost-of-fixing-software-security.htm
3. Allen, J. H., et al. (2008) Software security engineering: a guide for project managers, Addison Wesley, 2008

4. Jacobson, I (1992) Object oriented software engineering: use case driven approach, Addison Wesley
5. Kotonya, G and Sommerville, I (1998) Requirements Engineering: Processes and Techniques, Wiley.
6. Lamsweerde, van A (2009) Requirements Engineering: From system goals to UML models to software specifications, Wiley, UK.
7. Sommerville, I and Sawyer, P (1998) Requirements Engineering: A good practice guide, Wiley.
8. Firesmith, D (2007) Engineering Safety- & Security-Related Requirements ICCBSS Tutorial, SEI, Carnegie Mellon University, 27 February.
9. Firesmith, D (2003) Engineering security requirements, Journal of Object Technology, Volume 2, No. 1, 2003
10. CERT-SEI, www.cert.org
11. CERT-UK, https://www.cert.gov.uk/
12. BSI (2013) Attack patterns articles, https://buildsecurityin.us-cert.gov/articles/knowledge/attack-patterns
13. Schneier, B (1999) Attack Trees: modelling security threats, Dr Dobbs Journal, December, http://www.schneier.com/paper-attacktrees-ddj-ft.html
14. Schneier, B (2000) Secrets and Lies: Digital Security in a Networked World. New York, NY: John Wiley & Sons
15. Ellison, R.J. and Moore, A. P (2003) Trustworthy Refinement Through Intrusion-Aware Design (CMU/SEI-2003-TR-002, ADA414865). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
16. Howard, M and LeBlanc, D. C (2002) Writing Secure Code (2nd ed.). Redmond, WA: Microsoft Press.
17. Mead, N. R et al. (2008) Incorporating Security Quality Requirements Engineering (SQUARE) into Standard Life-Cycle Models, SEI Technical Note CMU/SEI-2008-TN-006, http://www.sei.cmu.edu
18. Caralli, R. A et al. (2007) Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process, TECHNICAL REPORT, CMU/SEI-2007-TR-012
19. Alberts, C and Dorofee, A (2002) Managing Information Security Risks: The OCTAVE^{SM} Approach, Addison Wesley
20. Woody, C and Alberts, C (2007) Considering Operational Security Risk during System Development", C Woody, C Alberts, IEEE Security & Privacy, pp. 30-43
21. CLASP (2006) OWASP CLASP Version 1.2, http://www.lulu.com/items/volume_62/1401000/1401307/3/print/OWASP_CLASP_v1.2_for_print_LULU.pdf
22. S-SDLC: Introducing Secure Software development Life Cycle (S-SDLC), Infosec Institute, http://resources.infosecinstitute.com/intro-secure-software-development-life-cycle/
23. Ramachandran, M (2012) Software Security Engineering: Design and Applications, Nova Science Publishers, New York, USA, 2012. ISBN: 978-1-61470-128-6, https://www.novapublishers.com/catalog/product_info.php?products_id=26331
24. Chen, A. Jia (2004) Security engineering for software (SES), CS996-CISM, isis.poly.edu/courses/cs996-management/Lectures/SES.pdf
25. Belapurkar, A., et al. (2009) Distributed system security: issues, processes and solutions, Wiley.
26. Ramachandran, M., Chang, V., and Li, C-S (2015) The Improved Cloud Computing Adoption Framework to deliver secure services, Emerging Software as a Service and Analytics -

ESaaSA 2015 in conjunction with 5th International Conference on Cloud Computing and Services Science - CLOSER 2015, http://closer.scitevents.org/ESaaSA.aspx
27. Ramachandran, M (2014) Enterprise Security Framework for Cloud Data Security, Book chapter "Delivery and Adoption of Cloud Computing Services in Contemporary Organizations, Chang, V (ed.) IGI GlobalGraham, D (2006) Building Security In, https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/548-BSI.html