Security Software Engineering: Do it the right way

Ahmad AlAzzazi,
Faculty of Information Systems & Technology
Arab Academy for Banking and Financial Sciences
Amman, Jordan

Asim El Sheikh,
Faculty of Information Systems & Technology
Arab Academy for Banking and Financial Sc.
Amman, Jordan

Abstract: - Secure software development is one of the most information system issues that raised through the use of the internet and networked systems. The importance of developing secure software increases. In this work we present a process for the development of security critical software projects and an overview of some of the existing processes, standards, life cycle models that support the secure software development. It is a guide to the common body of knowledge for producing, acquiring, and sustaining secure software.

Key-Words: - Software engineering for Secure Systems, Software Engineering, SSE-CMM, Security Quality Requirements, and Engineering (SQUARE)

1 Introduction

With the wide use of the Internet, and increasing need of e-commerce solutions, e-banking, etc., the risks from malicious attacks are increasing. The need not only for physical protection of the software systems, but there is also a need to think about the process of building the software for security critical systems.

Software engineering is concerned with the use of engineering principles for developing quality software in a predictable way, engineering is concerned with all aspects of software production that integrates the processes “a framework for software development” [3]. There are general software engineering processes developed over the past decade.

One of the real challenges facing the emerging field of software engineering is security. There is a lack of an easily accessible common body of knowledge in this area. Simply put most software developers and architects, the very people who need to understand and practice software security, remain blithely unaware of their critical role.

Due to a report by the Department of Trade and Industry in the UK 60% of organizations have suffered security breaches in the last two years but only 37% of organizations undertake a risk assessment identifying critical assets and 40% of companies that have experienced serious security breaches still do not have any contingency plans to deal with future attacks[11].

Threats from a software security breach could range from the very mild (such as the defeat of copy protection) to the disastrous.

US Department of Defense (DoD) found that 88% of their computers were penetrable, and 96 % of those did not notice penetration [11].

Software system designers today must think not only of users, but also of all threats. Security concerns must inform every phase of software development, from requirements engineering to design, implementation, testing, and deployment, which means that security couldn't be handled as a side issue.

This paper suggests that techniques must be developed and used to build trust in software tools and processes. This leads to an interaction between software engineering and security engineering, which rise to give to several fascinating research challenges and opportunities. We suggest that security requirements must be incorporated into software systems today. In this paper we argue that security cannot be a side issue of a software project, it is a core of the project, which couldn’t function without it.

2 Security Engineering and Software Engineering

Devenbau says: Security, like beauty, is in the eye of the beholder [1]. A public library will clearly have a different view of computer security than will a bank transaction.

Security is a Non-functional emergent property of a system, like reliability, performance and safety. It could be defined as the system attribute that reflects the ability of the system to protect itself from external attacks that may be accidental or deliberate, the concealment of information or resources, the

prevention of unauthorized disclosure of information, the extent that the software itself must be hidden or obscured, the trustworthiness of data or resources [3].

The three Properties of Security are Confidentiality, Integrity and Availability, with other extended Properties like Accountability and Authenticity.

A security system consists of hardware, software, people, procedures and culture. There exists many security enforcements like the Network Security, Computer Security, Application Security and Software Security.

Secure software is software that cannot be intentionally forced to perform unintended functions; it is the process of designing, building and testing software for security [5]. This gives the interaction between Security Engineering and Software Engineering. Where software engineering is the use of engineering principle to developing quality software in a predictable way, engineering is concerned with all aspects of software production that integrates the processes, Methods and the Tools. For a Software Engineer the following question must be answered: How do we design, build, verify, and maintain software to be secure? This leads to the point that Software security includes software architectures and structures to improve security, techniques and tools that help with reasoning about software security.

Software security is concerned with algorithm design, traditional cryptography, education, design, interface selection, specification, Implementation and coding, verification, testing and evaluation, deployment and secure execution, maintenance and bug fixing, and refinement. This means that building secure software is a process that ends with an artifact; and software security is concerned with all phases of a software engineering process.

We could argue that current development practice suffers from different key problems like, Security requirements tend to be kept separate from other system requirements, and not integrated into any overall strategy [6].

The overwhelming majority of existing software, even software for safety-critical, security-critical systems, has been built and is being built in an ad hoc, unsystematic fashion. In our work, we built a frame work that defines the security activities that have to be done in all the phases of the software engineering process from the specification of the requirements to the deployment with a deep look in each phase for the security activities that have to be done.

3 Major Components of Security Engineering

Before looking into security activities in each phase of our security process (next section), we have a short look at the major components of security engineering, which are almost unknown for a software engineer. They are important to determine the software's Security needs. First, one has to define the security goals based on actual needs, and only then review the entire system and security policies with the goals in mind.

Steps for making a secure system are [2]:

1- Security Assessment
2- Security Design
3- Security Implementation
4- Security Monitoring

3.1 Security Assessment

Conducting Security Assessment provides detailed recommendations for identifying, correcting and preventing security problems. The following issues are involved: Asset Identification, Threat Assessment, Laws, Regulation and Policies, Personnel, Reporting and Follow-up, Tools, and Training Certification.

3.2 Security Design

The Security Components are divided into two main categories, the physical and logical parts. Physical Security includes elements like Buildings, Devices (e. g. Network Devices, Equipment, Cables, Communication devices, Servers, Routing Equipment and other Hardware devices), and Human (internal and external). On the other hand, the logical Security includes the Application Security, Operational Security, systems, Acceptable Use Policy (AUP), Intrusion detection /prevention /recovery, use procedures, authentication, identification, privacy, integrity, non-repudiation, encryption Backups, Risks, Assessment, access control list (ACL), Access Cards, Virtual Private Network (VPN) and Biometric Authentication.

3.3 Security Implementation

For all of these components, there are a lot of policies from a Security engineering perspective that have to be considered.

3.4 Security Monitoring

There is much more to security analysis than code review. There are some Methods for performing that, like architectural security analysis, attach trees, source code auditing and others [10].

4 Software Engineering for Security

The vast majority of software developers continue to focus on code. Software security is fundamentally a software engineering problem, encompassing producing and evaluating secure software. Developing secure software needs support from all aspects of software processes, methods, and tools. A majority of security incidents result from defects in software requirements, design, or code. Specification errors can be easier and cheaper to fix in the early stages – using analysis tools.

How is Security Viewed in Software Engineering? It is a Non-functional requirement, an Emergent property, an Aspect of dependability, to use security mechanisms for the implementation of the security policies.

We must engineer security into software as an up front requirement rather than a last minute thought to recognize that exploits/vulnerabilities are results of flawed design or implementation.

The overwhelming majority of existing software, even software for security-critical systems, has been built and is being built in an ad hoc, unsystematic fashion.

We have to consider carefully the security aspects of the software product from the beginning with requirements and moving on through later lifecycle activities, ending with deployment and administration of the software product.

4.1 Requirements for a secured system

The requirements of a specific security system can only be determined after detailed consideration of the business context, user preferences, and the defense posture of the software system.

A security requirement is a manifestation of high-level policies into the detailed requirements of a specific software system. The security specification process which leads to the early stage of the software process lifecycle is shown in figure 1 [3].

The following are the Stages of security specification: A) Asset identification and evaluation; which assets (data and programs) and their required degree of protection are identified. The degree of required protection depends on the asset value so that a password file (say) is more valuable than a set of public web pages. B) Threat analysis and risk assessment, the possible security threats are identified and the risks associated with each of these

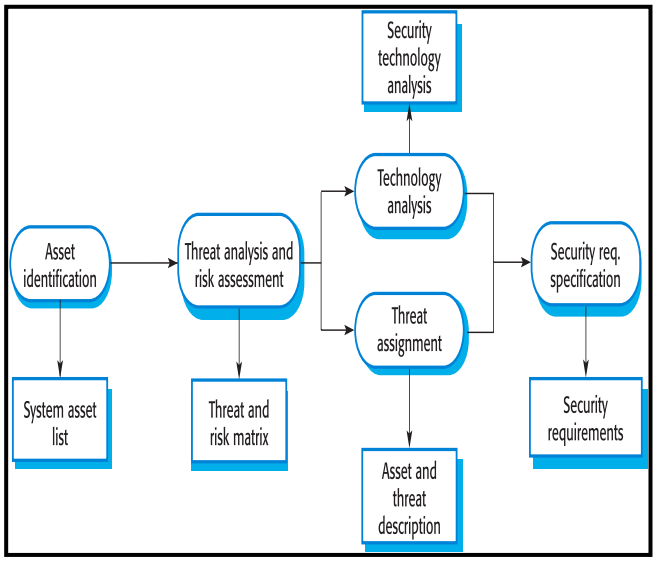

Figure 1: specification process of security

threats is estimated. C) Threat assignment; the identified threats are related to the assets so that, for each identified asset, there is a list of associated threats. D) Technology analysis; the available security technologies and their applicability against the identified threats are assessed. E) Security requirements specification; the security requirements are specified. Where appropriate, these will explicitly identify the security technologies that may be used to protect against different threats to the system. They include the following types of security requirements: Identification requirements, Authentication requirements, Authorization requirements, Immunity Requirements, Integrity Requirements, Intrusion detection requirements, Non-repudiation requirements, Privacy requirements, Security auditing requirements, and system maintenance security requirements.

4.2 Modeling for a Secured System

The second design phase focuses on clarifying the model of the system and the security requirements. Dependencies between the assets of the system must also be identified based on the information gathered in the security requirements phase to select or design the appropriate analysis and security design.

For the security analysis and security design part of the process, it is important to ensure that expert knowledge is available in order to identify threats and countermeasures. Designers of secure software systems can have a number of distinctive goals including: design to defend after initial security

violation(s), architecturally eliminate possibilities for violations.
Some of the architectural styles or style elements could include: Reference monitors, Multiple Independent Levels of Security (MILS), Multiple Single Levels of Security (MSLS), Distributed access control, Tolerant – to (partial) attacker success – including "self healing" approaches, Adaptive distributed reconfiguration responses to attacks, Compartmentalization via Virtual machines, Separation via encryption, Physical separation, Separation except at point of use Filters, guardians, and firewalls.

4.3 Construction for a Secured System

Secure software construction creates working, meaningful, secure software through design, coding, verification, unit testing, integration testing, and debugging [5]. Some vulnerabilities in software systems occur with such great frequency they have been deemed "common vulnerabilities." To effectively avoid vulnerabilities, one must fully understand the types of weaknesses that enable them. We have to use Security Principles in Secure Coding like Input Validation, Preventing Buffer Overflow, Anti-Tamper Technologies, and the use Secure Coding Standards, etc.

Our Research Model should describe the detailed security activities that have to be done in each phase of the development of the security critical software product. In the following two sections of this paper, we show two different approaches dealing with security critical software development.

5 Systems Security Engineering Capability Maturity Model (SSE-CMM)

In comparison to our work, we show the SSE-CMM, which is a process model that can be used to improve and assess the security engineering capability of an organization [7]. The SSE-CMM provides a comprehensive framework for evaluating security engineering practices against the generally accepted security engineering principles. By defining such a framework, the SSE-CMM, provides a way to measure and improve performance in the application of security engineering principles. The SSE-CMM has been adopted as the ISO/IEC 218. The stated purpose for developing the model is that, although the field of security engineering has several generally accepted principles, it lacks a comprehensive framework for evaluating security engineering practices against the principles. The SSE-CMM, by defining such a framework, provides a way to measure and improve performance in the application of security engineering principles. The SSE-CMM also describes the essential characteristics of an organization's security engineering process. The model is organized into two broad areas: Security Engineering, and Project and Organizational processes. Security Engineering in turn is organized into Engineering Processes, Assurance Processes, and Risk Processes. There are 22 Process Areas distributed among the three categories. Each Process Area is composed of a related set of process goals and activities [7]. The International Systems Security Engineering Association (ISSEA) maintains the SSE-CMM (Figure 2).

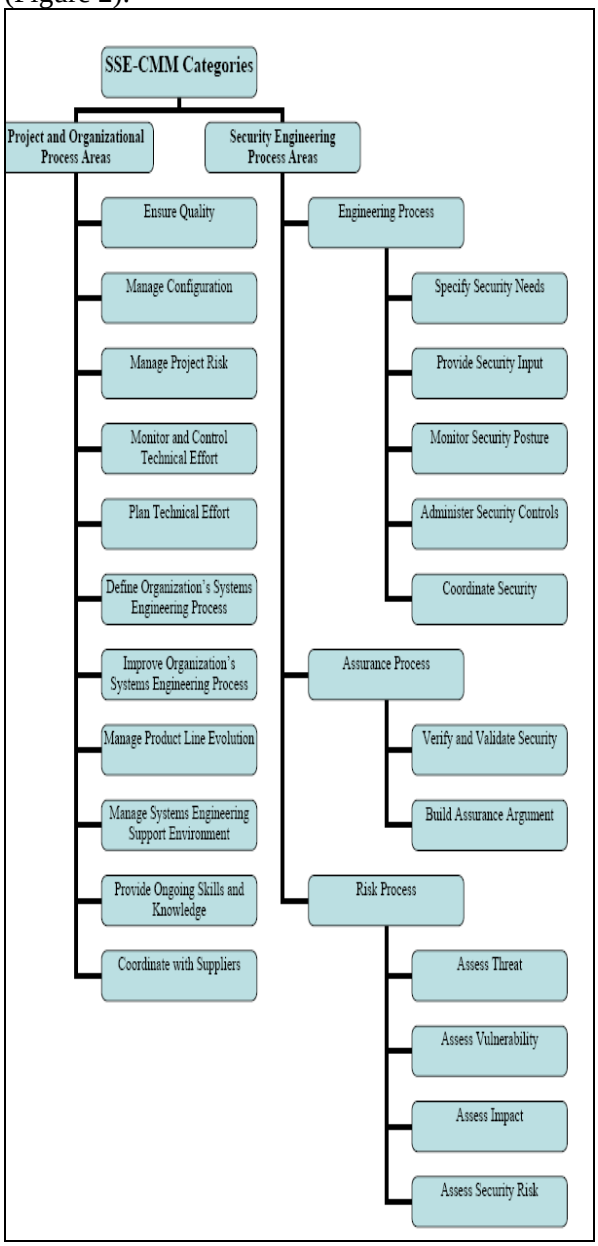

Figure 2: The SSE-CMM

The SEE-CMM is a straightforward analysis of the existing processes to determine which base process have met and the maturity levels they have achieved, which is not simple and may involve interactions with engineers who actually use the process.

6 Security Quality Requirements Engineering (SQUARE) Methodology

Another comparable methodology to our work is the Security Quality Requirements
Engineering (SQUARE) Methodology for eliciting and prioritizing security requirements in software development projects, which was developed by the Software Engineering Institute's Networked Systems Survivability (NSS) Program [8]. The methodology's steps are explained, and results from its application in recent case studies are examined. The Methodology consists of 9 steps to elicitate the requirements (step 1: Agree on Definitions, Step 2: Identify Security Goals, Step 3: Develop Artifacts, Step 4: Perform Risk Assessment, Step 5: Select Elicitation Technique, Step 6: Elicit Security Requirements, Step 7: Categorize Requirements, Step 8: Prioritize Requirements, Step 9: Requirements Inspection). Each step is evaluated using Inputs, the used Techniques, the Participants and the Outputs of the step.

The SQUARE) Methodology concentrates on the requirement phase in software development, which is only a part of the overall process.

7 Conclusion

The Development of secured software gives us the interaction between two disciplines, software engineering and security engineering. At a software engineering process we have to consider the security aspects in all of the phases of the Software development process from specification of the requirements until construction. This consideration of all of the security aspects should not be an ad hoc solution to the software problem (as in most cases).
Some standards exits in the area of the development of secured systems, but many research areas still need to be discussed. The need for more secured systems urges us to look deeper into software engineering processes to build better frameworks for them. That means, there is growing need to incorporate security engineering into standard analysis and design processes and security requirements must not be left to be dealt with as a side-issue, or an afterthought. Some existing methodologies are looking either on existing process like the SEE-CMM or concentrating only on some parts of the software engineering process like the (SQUARE) Methodology.
In our work, a framework for all of the software engineering activities should be built with a deep consideration of the security activities in each phase.

References:

[1] Devanbu, T & Stubblebine, S., Software Engineering for Security: A Roadmap, Proceedings of the 22nd International Conference on Software Engineering, 2000,Pages: 227 - 239.

[2] Estublier, J. Software configuration management: a roadmap, Proceedings of the 22nd International Conference on Software Engineering, 2000, Pages: 279 - 289.

[3] Sommerville, I., Software Engineering, 6th Edition, Prentice-Hall, 2000.

[4] Redwine, S., Secure Software Assurance, US Departments of Homeland Security and Defense, https://buildsecurityin.us-cert.gov, 2006, accessed 10/09/2006.

[5] Abrams, M. D., Security Engineering in an Evolutionary Acquisition Environment. New Security Paradigms Workshop, Proceedings of the 1998 workshop on New security paradigms, Virginia, United States, 1998, Pages: 11 - 20

[6] Flechais, I, Angela, M., Sasse, S. & Hailes, M., Bringing Security Home:A process for developing secure and usable systems, Proceedings of the 2003 workshop on New security paradigms, Ascona, Switzerland, 2003, Pages: 49 - 57

[7] Software Engineering Institute at Carnegie Mellon (SEI) , "Technical Program Report", http://www.sse-cmm.org, 2003, accessed 21/07/2006.

[8] Mead, N., Hough, E. & Stehney, T., SEI,TECHNICAL REPORT CMU/SEI-2005-TR-009, http://www.sse-cmm.org, 2005,accessed 12/08/2006. .

[9] Jurjens, J., Secure Systems Development with UML, Springer Verlag-Berlin, 1998.

[10] Viega, J. & McGraw, G, How to avoid Security Problems the right way, Addison-wesley, 2002.

[11] Information Security Breaches Survey, Technical Report, Department of Trade and Industry, www.pwc.com/uk/eng/ins-sol/publ, 2000, accessed 05/12/2006