SecSDM: A Model for Integrating Security into the Software Development Life Cycle

Lynn Futcher, Rossouw von Solms

Centre for Information Security Studies,
Nelson Mandela Metropolitan University,
Port Elizabeth, South Africa
{Lynn.Futcher, Rossouw.VonSolms}@nmmu.ac.za

Abstract. Most traditional software development methodologies do not explicitly include a standardised method for incorporating information security into their life cycles. It is argued that security considerations should provide input into every phase of the Software Development Life Cycle (SDLC), from requirements gathering to design, implementation, testing and deployment. Therefore, to build more secure software applications, an improved software development process is required. The Secure Software Development Model (SecSDM), as described in this paper, is based on many of the recommendations provided by relevant international standards and best practices, for example, the ISO 7498-2 (1989) standard which addresses the underlying security services and mechanisms that form an integral part of the model.

Keywords: Risk analysis, secure software development, security mechanisms, security services.

1 Introduction

It is within highly integrated technology environments that information security is becoming a focal point for designing, developing and deploying software applications. Ensuring a high level of trust in the security and quality of these applications is crucial to their ultimate success. Therefore, information security has become a core requirement for software applications, driven by the need to protect critical assets and the need to build and preserve widespread trust in computing.

A Microsoft study demonstrated that 64% of software developers are not confident in their ability to write secure applications [1]. This is despite the fact that the .Net framework and Visual Studio.Net provide them with the necessary tools and information to write secure applications. Software developers often rely on their intuition in developing secure software and do so without much systematic help or guidance. These professionals need be educated to put security at both the heart of software design and at the foundation of its development process [2]. This implies that software developers need to use improved processes that consistently produce secure software. Currently, no software development processes or practices exist that consistently produce secure software [3]. It is therefore recommended that software producers adopt

Please use the following format when citing this chapter:

Futcher, L., von Solms, R., 2007, in IFIP International Federation for Information Processing, Volume 237, Fifth World Conference on Information Security Education, eds. Futcher, L., Dodge, R., (Boston: Springer), pp. 41–48.

practices that can measurably reduce software specification, design and implementation defects and, therefore, minimise any potential risk.

This paper describes a model for integrating security into the SDLC. It proposes a more stringent software development methodology that both detects and removes vulnerabilities early in the life cycle, thereby minimising the number of security vulnerabilities in the live system. The SecSDM aims to draw attention to the importance of security in the SDLC. It is designed as an extension, not a replacement, to pre-existing software development methodologies.

2 The Software Development Life Cycle

The development of software has always been regarded as a difficult task. For this reason, many different methodologies have been proposed by various researchers to guide the software development process as a whole. Software development typically follows a life cycle which determines the phases along which the software product moves. The traditional SDLC is a methodology for the design and implementation of an information system in an organisation. There are many representations of the SDLC, all showing a logical flow of activity from the identification of a need to the final software product. Each methodology has its own strengths and weaknesses and is therefore well-suited for certain types of applications. Although more complex systems require more iterative development models, the five phases comprising the traditional linear SDLC model are inherent in most software development methodologies and therefore form the basis for the SecSDM.

If security considerations were woven into the SDLC, many of the security vulnerabilities that manifest themselves in live systems today would never appear [4]. Therefore, it is argued that a more stringent software development process that incorporates security is required. Such a process should minimise the number of security vulnerabilities present in the SDLC and detect and remove these vulnerabilities as early in the life cycle as possible [5]. New ways of addressing and resolving security issues, early within the SDLC, must be introduced in the software development arena [6].

Although many researchers advocate that security needs to be integrated into the SDLC, few are able to describe a process to achieve this goal. It is argued that the evident separation between information security and software development has resulted in the production of vulnerable software applications. Therefore, it is necessary to develop an improved software development process to build more secure software. Security concerns must provide input into every phase of the SDLC.

3 The Secure Software Development Model

Various international standards and best practices were consulted when developing this model. These include ISO/IEC 17799 [7], the international code of practice for information security management, the guidelines of the detailed risk analysis approach as determined by ISO/IEC TR 13335-3 [8], the NIST SP 800-14 [9] which

outlines generally accepted principles and practices for securing information technology systems, and ISO 7498-2 [11] which provides the basis of information security in software systems through five basic security services, supported by eight security mechanisms.

An important consideration in developing this model was to define a useable process that will lessen the burden for software developers who are not specialists in information security. This section describes the SecSDM as a simple, ten-step process for integrating security concerns into each phase of the SDLC as follows:

- *Investigation Phase*: determines the security requirements of the software application by executing a simple risk analysis exercise;
  - STEP 1: Information asset identification and valuation;
  - STEP 2: Threat identification and assessment;
  - STEP 3: Risk (asset/threat) identification;
  - STEP 4: Determine the level of vulnerability;
  - STEP 5: Risk assessment;
  - STEP 6: Risk prioritisation.
- *Analysis Phase*: determines the security services to be used to satisfy the security requirements;
  - STEP 7: Identify the relevant security services and level of protection required to mitigate each risk.
- *Design Phase*: determines how the security services will be implemented;
  - STEP 8: Map security services to security mechanisms;
  - STEP 9: Consolidate security services and mechanisms.
- *Implementation Phase*: identifies and implements appropriate software security tools and components;
  - STEP 10: Map security mechanisms to software security components.
- *Maintenance Phase*: the maintenance of software is made easier and more manageable through the structured approach provided by the SecSDM. Users and operations staff need to be educated in using the software application in a secure manner.

3.1 The investigation phase

Early determination of security requirements is necessary to develop software applications which can be trusted by all stakeholders. ISO/IEC TR 13335-3 [8] suggests that information security requirements are stated in terms of confidentiality, integrity, availability, accountability, authenticity and reliability of information. Therefore, it is necessary to perform some form of a risk analysis to determine the security requirements of a particular system.

The proposed risk analysis approach carried out during the investigation phase takes the form of a step-by-step process. Its purpose is to identify the information assets, their associated threats and vulnerabilities, and rank them according to those assets that need the most protection. Different industries and different systems have varying information protection requirements. For example, healthcare organisations stress the confidentiality of patient records, whereas banking is more concerned about

the integrity of monetary transactions. The software development team needs to understand and capture what the adequate protection of information is, in their specific context [11].

STEP 1: Information asset identification and valuation

The listing of assets based on checklists and judgment, yields an adequate identification of the important assets associated with the software application being developed [12]. These information assets can include, for example, personal information, employee salary information, customer contact information or financial information.

The next step in the process is to assign values to each of the key information assets identified. This is necessary to determine the impact value and sensitivity of the information in use, stored, processed or accessed. The SecSDM uses a 5-point Lickert scale and requires that an asset impact value between 0 and 4 (where 0=negligible and 4=critical) be assigned to each of the key information assets identified. These values represent the business importance of the assets and will typically be obtained by interviewing the information owners and its key users. The next step requires the identification of the various threats that may cause harm to these assets.

STEP 2: Threat identification and assessment

It is necessary to perform the identification and assessment of threats during the investigation phase of the SDLC. This information is required to identify risks and to guide subsequent design, coding and testing decisions.

A checklist of the most common threats is provided by the SecSDM, based on those referred to in ISO/IEC TR 13335-3 [8]. Such a checklist of the most likely threats is helpful in performing a threat assessment, although software developers must be aware that threats are continually changing. Furthermore, it is necessary, as part of the threat assessment process, to determine the potential impact that each of the common threats may have on the assets associated with the software application. This may be performed, according to the SecSDM, by assigning each of the threats identified to one of the likelihood levels (low, medium or high).

STEP 3: Risk (asset/threat) identification

Risk identification requires that the most critical asset/threat relationships are identified to ascertain which risks are most likely to impact the proposed system [13]. This is done by simply considering the key information assets, as identified in Step 1, and the most likely threats identified in Step 2. Those assets with high or critical asset impact values (i.e., 3 or 4) and those threats recognised to have a potentially high impact will contribute significantly to the criticality of the risk. The following step in the process requires that the level of vulnerability for each critical risk be determined.

STEP 4: Determine the level of vulnerability

In practice, security is not compromised by breaking the dedicated security mechanisms, but by exploiting the weaknesses or vulnerabilities in the way they are used [2]. Therefore, as part of the risk analysis process, it is important to be able to determine the level of vulnerability for each risk. It is necessary to consider the likelihood that the risk may materialise, taking the current situation and controls into account, to

determine the level of weakness or vulnerability for each risk. The three main levels of vulnerability provided by this model are low, medium and high. The following step in the risk analysis process requires that a risk assessment be carried out to determine the extent of each risk.

STEP 5: Risk assessment

The extent of risk is determined, according to the SecSDM, by taking into account the asset impact value, level of vulnerability and potential likelihood of each threat identified. These are matched in a lookup table to establish the specific measure of risk on a scale of 1 to 8. The specific risk values established are determined according to those recommended by ISO/IEC TR 13335-3 [8].

STEP 6: Risk prioritization

The prioritisation of risks during the investigation phase serves as a guideline for the analysis, design and implementation phases of the SDLC. This is achieved by simply listing each risk, identified in Step 3, and its corresponding risk value as established in Step 5.

3.2 The analysis phase

The risk extent of a particular software application determines the scope of the security services employed. Therefore, it is meaningful for the analysis phase to focus on the security risks as identified during the investigation phase. During the analysis phase, security services are selected according to their ability to mitigate the security risks identified. It is important, however, that this is carried out independently of any implementation details. The output of this phase is a refined set of security requirements.

The ISO 7498-2 standard provides the basis for information security in software applications through five basic security services, namely: identification and authentication, authorisation/access control, confidentiality, integrity and non-repudiation/non-denial. These five security services provide the basis for ensuring the security of any software application [10].

STEP 7: Identify the relevant security services and level of protection required to minimise each risk

Software developers are required to map each of the most critical risks, as identified during the investigation phase, to the envisaged security services. For each risk, multiple security services may be identified. However, not all security services are required to address each individual risk, and neither are all security services applicable to all risks. This step results in the appropriate level of protection being selected to reduce the risks to an acceptable level. The next section describes the process of selecting the appropriate security mechanisms through which the security services, identified in Step 7, should be implemented.

3.3 The design phase

It is during the design phase of the SecSDM that the security services need to be translated into security mechanisms. The five security services referred to by ITU-T X.800 and the ISO 7498-2 standard are supported by eight security mechanisms, namely: encipherment, digital signatures, access control, data integrity, authentication exchange, traffic padding, routing control and notarisation [10]. These security mechanisms, however, cannot be "blindly" inserted into a software application in the hope of providing the required level of security. The overall system development process needs to take the various security concerns and risks into consideration to ensure the appropriate use of the required security mechanisms.

STEP 8: Map security services to security mechanisms

The SecSDM provides guidelines to assist software developers in selecting the most appropriate security mechanisms to support the security services identified during the analysis phase. For example, if confidentiality is a required security service, then encryption can be used as the security mechanism. The mapping of security services to the appropriate security mechanisms is required for all risks identified during the investigation phase.

STEP 9: Consolidate security services and mechanisms.

For ease of implementation, the SecSDM requires the consolidation of the results of Steps 7 and 8. Software developers are required to map the various security mechanisms to the appropriate security services for each risk, as identified during the investigation phase.

3.4 The implementation phase

During the previous phases, according to the SecSDM, the risk sensitivity of the system has been determined and the most appropriate security services and mechanisms to be employed have been identified. These mechanisms need to be implemented. The implementation of security mechanisms depends on the programming language used, the coding standards and best practices adhered to, and the personal programming style of the programmer. It is important to ensure that developers are knowledgeable about security risks and skilled in secure coding standards [4]. The programmer must ensure that all security-relevant code is understandable, auditable, maintainable and testable [14].

An important part of the implementation phase is testing. Testing is often seen as a way of 'testing in' security which is unacceptable. The role of security testing is to verify that the system design and code can withstand attack. Testing ensures that countermeasures are correctly implemented and that code is developed following coding standards and best practices. Security testing should follow a security test plan. This test plan should include unit testing, integration testing, quality assurance testing and penetration testing [4]. The testing of the software to validate that it meets the se-

curity requirements as determined during the investigation phase is essential to produce secure software. This testing should include serious attempts to attack and break its security and scan for common vulnerabilities [3].

STEP 10: Map security mechanisms to software security components.

The security mechanisms identified may be implemented through appropriate software security tools and components, for example, those inherent in the .Net framework. The .Net framework provides developers with the necessary tools and information to write secure applications [1]. The SecSDM does not currently recommend the use of specific software security components to implement the various security mechanisms. However, it does describe the process of mapping the security mechanisms summarised in Step 9 to various software security components as recommended by the software developer. An implementation priority list is needed which indicates the priority of the security mechanisms to be implemented to ensure that the correct security features are employed [15]. Therefore, software developers are encouraged to indicate the specific software security components through which the various security mechanisms will be implemented.

3.5 The maintenance phase

The maintenance phase is often viewed as another iteration of the entire life cycle [16]. During this phase, it is important to find ways to evaluate the security of the system to ensure that the system is as secure as intended. The SecSDM ensures that all relevant security-related information is well documented. This helps improve the auditability of the software application in question, because security-related decisions are traceable to the appropriate phase as proposed by this secure software development approach. The integration of information security into the SDLC as described in this section, and the tight integration between the various phases will help ensure that the final product meets the information security requirements, identified during the initial phases.

4 Conclusion

By applying the SecSDM, security is tightly interwoven in the software development process. Software developers are encouraged to consider security from the earliest phases of the SDLC, and to build critical security milestones and events into their development timelines. The concepts from each phase of the SecSDM should be integrated into the corresponding phases of the existing SDLC to ensure that security is appropriately considered and built into the software application. This type of inclusion should result in a robust end product that is more secure, easier to maintain and less costly to own.

The SecSDM, as described in this paper, has been implemented with an associated methodology at a tertiary institution. Initial experiments have shown encouraging re-

sults. A further positive aspect is that the associated documentation ensures that the entire security analysis and implementation process is auditable.

References

1. Taft, D. K. (2004, Dec). Microsoft aids secure coding. eWeek.
2. Jurjens, J. (2002, May). Using UMLSec and goal trees for secure systems development. Communications of the ACM, 48 (5), pp.1026-1030.
3. Task Force Report. (2004, April). Improving security across the software development life cycle (Technical Report). National Cyber Security Summit.
4. Jones, R. L.& Rastogi, A. (2004, Nov). Secure coding - building security into the software development life cycle. Application Program Security, pp.29-38.
5. Lipner, S.& Howard, M. (2005). The trustworthy computing security development lifecycle. 27. (cited on 15th April 2005)
6. Tryfonas, T.& Kiountouzis, E. (2002). Information systems security and the information systems development project. In Proceedings of IFIP.
7. ISO. (2005). ISO/IEC 17799 : Information Technology - Code of Practice for Information Security Management.
8. ISO. (1998). ISO/IEC TR 13335-3 : Information Technology – Guidelines for the Management of IT Security. Part 3 : Techniques for the management of IT security.
9. NIST (1996, Sept). Generally accepted principles and practices for securing information technology systems. *NIST Special Publication 800-14*. (http://csrc.nist.gov/publications)
10. ISO. (1989). ISO 7498-2: Information Processing Systems - Open System Interconnection - Basic Reference Model - Part 2: Security Architecture.
11. Tipton, H. F.& Krause, M. (2006). Information security management handbook (Fifth ed., Vol. 3). New York : United States of America: Auerbach Publications.
12. Landoll, D. J. (2006). The security risk assessment handbook : A complete guide for performing security risk assessments. New York : United States of America: Auerbach Publications.
13. Whitman, M.& Mattord, M. (2003). Principles of information security. Thomson Course Technology.
14. Tompkins, F. G.& Rice, R. (1985). Integrating security activities into the software development life cycle and the quality assurance process. In Proceedings of IFIP pp.65-105.
15. Siponen, M., Baskerville, R.& Kuivalainen, T. (2005). Integrating security into agile development methods. In Proceedings of the 38th Hawaii international conference on system sciences.
16. Gregory, P. H. (2003). Security in the software development life cycle (Technical Report). The Hart Gregory Group Inc.