Deployment Package
Design with Security Considerations
Basic Profile + Security

Notes:

Authors	Perla Maciel – CIMAT A.C. Jezreel Mejía – CIMAT A.C. (México)
Editors	Perla Maciel – CIMAT AC. (México) Jezreel Mejía – CIMAT A.C. (México)
Creation date	24/06/21
Last update	24/06/21
Version	0.1

Version History

Date	Version	Description	Author
2009-07-08	0.1	Document creation	Perla Maciel

Abbreviations/Acronyms

Abre./Acro.	Definitions
DP	Deployment Package - a set of artefacts developed to facilitate the implementation of a set of practices, of the selected framework, in a Very Small Entity.
VSE	Very Small Entity – an enterprise, organization, department or project having up to 25 people.
VSEs	Very Small Entities
TL	Technical Leader
AN	Analyst
DES	Designer
SA	Security Advisor

Table of Contents

1. Technical Description

Purpose of this document

This Deployment Package (DP) supports the Basic Profile as defined in ISO/IEC 29110 Part 5-1-2: Management and engineering guide[1]. The Basic Profile is one profile of the Generic profile group. The Generic profile group is composed of 4 profiles: Entry, Basic, Intermediate and Advanced. The Generic profile group is applicable to VSEs that do not develop critical software. The Generic profile group does not imply any specific application domain.

A DP is a set of artefacts developed to facilitate the implementation of a set of practices in a Very Small Entity (VSE). A DP is not a process reference model (i.e. it is not prescriptive). The elements of a typical DP are: description of processes, activities, tasks, roles and products, template, checklist, example and reference to standards and models, and tools.

The purpose of this document, entitled "*Deployment Package - Software Architectural and Detailed Design*" is to provide VSEs with tailorable and easily usable guidelines and materials in order to implement a good *Software Design*.

The content of this document is entirely *informative*.

This document has been produced by Perla Maciel and Jezreel Mejía of CIMAT A.C. (México).

Why Design with Security Considerations?

The design phase brings a really good opportunity to avoid security holes introduced in all the software implementation process, which means that the rework can be prevented as well as the parching of the security holes after the delivery of the application. Define a security design phase or at least some practices to consider in the implementation

[2] ISO/IEC 29110-5-1-2 is available at no cost from ISO:
http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

2. Definitions

In this section, the reader will find two sets of definitions. The first set defines the terms used in all Deployment Packages, i.e. generic terms. The second set defines terms used in this Deployment package, i.e. specific terms.

Generic Terms

Process: set of interrelated or interacting activities which transform inputs into outputs [ISO/IEC 12207].

Activity: a set of cohesive tasks of a process [ISO/IEC 12207].

Task: required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process [ISO/IEC 12207].

Sub-Task: When a task is complex, it is divided into sub-tasks.

Step: In a deployment package, a task is decomposed in a sequence of steps.

Role: a defined function to be performed by a project team member, such as testing, filing, inspecting, coding. [ISO/IEC 24765]

Product: piece of information or deliverable that can be produced (not mandatory) by one or several tasks. *(e. g. design document, source code)*.

Artefact: information, which is not listed in ISO/IEC 29110 Part 5, but can help a VSE during the execution of a project.

Specific Terms:

Cryptographic design: Cryptographic technologies such as encryption, digital signatures, key management, and secret sharing schemes are critical elements in the implementation of any security service. *[Chan Yeob Yeun, Khalifa University]*

Threat Model: Steps to identify targets and vulnerabilities, identify measures to prevent or minimize the impact of threats on your system, and optimize the security of an application, system, or process Your Business. *[Michael Cobb]*

3. Tasks, Steps, Roles and Products in ISO/IEC29110

Process: Software Implementation (SI)

As defined in ISO/IEC 29110, the purpose of the Software Implementation process is the systematic performance of the analysis, design, construction, integration and tests activities for new or modified software products according to the specified requirements.

Activity: SI.3 Software Architectural and Detailed Design

The Software Architectural and Detailed Design activity transforms the software requirements to the system software architecture and software detailed design.

Task List	**Roles**
SI.3.3 Document or update the Software Design	**AN, DES**

Tasks

- **Satisfy Minimum Cryptographic Design Requirements**
- **Complete Threat Models**

Satisfy Minimum Cryptographic Design Requirements

Objectives:	Satisfy the minimal cryptographic design requirements established for your product when you established Security Requirements.
Roles:	AN – Analyst
	DES – Designer
	SA – Security Advisor
Products:	Software Design
	Requirement specifications
	Test Cases and Test Procedures
	Software Configuration
	Validation Results
Artifacts:	Minimum Cryptographic Design
Steps:	1. Identify all the Cryptographic Standards
	2. Select the Cryptographic Standards that apply to the project
	3. Document the minimum Cryptographic Standards to consider in the Design phase
Step Description:	**Step 1. Identify all the Cryptographic Standards**

	Identify all the Cryptographic Standards that can be introduced in the project, i.e. as the Microsoft Cryptographic Standards for SDL-covered products, at high-level are: • Use AES for symmetric enc/dec. • Use 128-bit or better symmetric keys. • Use RSA for asymmetric enc/dec and signatures. • Use 2048-bit or better RSA keys. • Use SHA-256 or better for hashing and message-authentication codes. • Support certificate revocation. • Limit lifetimes for symmetric keys and asymmetric keys without associated certificates. • Support cryptographically secure versions of SSL (must not support SSL v2). • Use cryptographic certificates reasonably and choose reasonable certificate validity periods. ***Step 2.* Select the Cryptographic Standards that apply to the project** After Identifying all the Cryptographic Standard that apply to the project, a minimum must be selected and specified as such. *Tip: Every language has their own libraries to implement the different Cryptographic Standards, and also their recommendation of which to use.* ***Step 3. Document the minimum Cryptographic Standards to consider in the Design phase*** The selected Cryptographic Standards must be documented and taken as the minimum in regard of security implementation. So, in the design phase that minimum is taken as a baseline a used in any data that requires a higher level of security.

Complete Threat Models

Objectives:	Execute a complete Threat models
Roles:	AN – Analyst
	DES – Designer
	SA – Security Advisor

Products:	Software Design
	Requirement specifications
	Test Cases and Test Procedures
	Software Configuration
	Validation Results
Artifacts:	Threat Models
Steps:	1. Create the Complete threat models for all functionality identified
	2. Ensure that all threat models meet minimal threat model quality requirements
	3. Have all threat models and referenced mitigations reviewed and approved
	4. Threat model documentation.
Step Description:	**Step 1. Create the complete threat models for all functionality identified.** Threat models typically need to consider the following areas: - All projects. Code exposed to the attack surface and code created or licensed by a third party. - New project. All functionalities and features. - Updated version of an existing project. New features added in the updated version. **2. Ensure that all threat models meet minimal threat model quality requirements.** All threat models should include data flow diagrams, assets, vulnerabilities, and mitigations. Threat modeling can be done in a variety of ways, including defining approaches using tools or documentation / specifications. **3. Have all threat models and referenced mitigations reviewed and approved.** Ask architects, developers, testers, program managers, and other users to understand the software contributing and review the threat model. Seek extensive feedback and reviews to ensure that our threat model is as complete as possible. **4. Threat model documentation** Threat model data and associated documentation (functional/design specs) should be stored using the document control system used by the product team.

Role Description

This is an alphabetical list of the roles, abbreviations and list of competencies as defined in Part 5-1-2.

	Role	*Abbreviation*	*Competency*
1.	Analyst	AN	Knowledge and experience eliciting, specifying and analyzing the requirements. Knowledge in designing user interfaces and ergonomic criteria. Knowledge of the revision techniques and experience on the software development and maintenance. Knowledge of the editing techniques.
2.	Designer	DES	Knowledge and experience in the software components and architecture design. Knowledge of the revision techniques and experience on the software development and maintenance. Knowledge of the editing techniques. Knowledge and experience in the planning and performance of integration and system tests.
3.	Security Advisor	SA	Knowledge to advise the businesses to identify potential security weaknesses, create security policies, and reduce risks to their IT systems.
4.	Technical Leader	TL	Knowledge and experience in the software development and maintenance.

Product Description

This is an alphabetical list of the input, output and internal process products, its descriptions, possible states and the source of the product.

IMPORTANT NOTE: Part 5.1.2 of ISO/IEC 29110 describes an example of Software Design content. This Deployment Package intentionally uses a different Software Design product content.

	Name	Description	Source
1.	*Requirements Specification*	Includes an introduction and a description of the development and security requirements. It may	Software Implementation

		contain: - Introduction –general description of software and its use within the scope of the customer business; - Requirements description: - functionality – established needs to be satisfied by the software when it is used in specific conditions. Functionality must be adequate, accurate and safe.	
2.	*Software Configuration*	A consistent set of software products including: - *Requirements Specification* - *Software Design* - *Traceability Record* - *Components* - *Software* - *Test Cases and Test Procedures* - *Test Report* - *Product Operation Guide* - *Software User Documentation* - *Maintenance Documentation* The applicable statuses are: delivered and accepted.	Software Implementation
3.	*Software Design*	This document includes textual and graphical information on the software structure and behaviour. This may include the following parts: Software Architectural Design – Describes the overall *Software* structure and behaviour: - Identifies architectural design stakeholders and their concerns - Identifies relevant architectural mechanisms (patterns, tactics, heuristics, rules of thumb, ...) - Identifies the types of views that are relevant to convey the software architecture, taking into consideration the stakeholders concerns and the various requirements (functional and non-functional) - Provides relevant software architectural views in various forms (diagrams, models, tables, plain text, ...) - Identifies and describes the main elements of the software architecture (subsystems, layers, modules) and their relationships - Identifies and describes the required software *Components,* their interfaces and the relationships among them - Describes rationale, provides any analysis	Software Implementation

		used to produce the solution, identifies known risks and inconsistencies. Detailed Software Design – includes details of the *Components* to facilitate their construction and testing within the programming environment: - Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) - Provides format of input / output data - Provides specification of data storage needs - Establishes required naming conventions - Defines the format of required data structures - Defines the data fields and purpose of each required data element - Provides the specifications of the program structure The applicable statuses are: verified and baselined.	
4.	*Test Cases and Test Procedures*	Test Case may include: - Identifies the test case - Test items - Input specifications - Output specifications - Environmental needs - Special procedural requirements - Interface dependencies Test Procedures may include: - Identifies: test name, test description and test completion date - Identifies potential implementation issues - Identifies the person who completed the test procedure - Identifies prerequisites - Identifies procedure steps including the step number, the required action by the tester and the expected results The applicable statuses are: verified and baselined.	Software Implementation
5.	*Traceability Record*	Relationship among the requirements included in the *Requirements Specification, Software Design* elements*, Components, Test Cases and Test Procedures.* - Identifies requirements of *Requirements Specification* to be traced - Provides forward and backwards mapping of requirements to *Software Design* elements,	Software Implementation

		Components, *Test Cases* and *Test Procedures*. The applicable statuses are: verified, baselined and updated.	
6.	*Validation Results*	May include the record of: - Participants - Date - Place - Duration - Validation check-list - Passed items of validation - Failed items of validation - Pending items of validation - Defects identified during validation	Project Management Software Implementation

Artefact Description

This is an alphabetical list of the artefacts that could be produced to facilitate the documentation of a project. The artefacts are not required by Part 5, they are optional.

	Name	Description
1.	Cryptographic Design	In this artefact a minimum of cryptographic considerations are established to help in the protection of the information.
2.	Threat Models	This models shows the threats that can put in danger the information managed by the software system developed.

4. References

Key	Reference
[CLEMENTS 03]	P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford, "Documenting Software Architectures - Views and Beyond", Addison Wesley, 512 pages, 2003, ISBN 0-201-70372-6.
[ISO/IEC 12207]	ISO/IEC 12207:2008 Systems and software engineering - Software life cycle processes.
[ISO/IEC 24765]	ISO/IEC 24765:2010, Systems and Software Engineering Vocabulary. (http://pascal.computer.org/sev_display/index.action)
[ISO/IEC 29110]	ISO/IEC 29110:2011, Software Engineering — Lifecycle Profiles for Very Small Entities (VSEs) — Part 5-1-2: Management and engineering guide – Generic profile group - Basic Profile.