Deployment Package
Construction with Security Considerations
Basic Profile + Security

Author	P. Maciel – CIMAT A.C. Jezreel Mejía – CIMAT AC. (México)
Editors	P. Maciel – CIMAT A.C. Jezreel Mejía – CIMAT AC. (México)
Creation date	29/06/11
Last update	25/06/21
Version	0.6

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

Version History

Date (yyyy-mm-dd)	Version	Description	Author
2011-06-29	0.5	Document Creation	Javier Flores
2021-06-25	0.6	Security Considerations	P. 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
PR	Programmer
PM	Project Manager

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

Table of Contents

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

1. Technical Description

Purpose of this document

This Deployment Package (DP) supports the Basic Profile as defined in ISO/IEC TR 29110-5-1-2:2011 Management and Engineering Guide. 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. The Basic Profile describes software development of a single application by a single project team with no special risk or situational factors.

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, reference and reference to standards and models, and tools.

The content of this document is entirely *informative*.

This document has been produced by Javier Flores (UNAM, México) and Ana Vazquez of 5th level (México) beyond her participation to ISO JTC1/SC7/WG24.

Why are Construction Security Consideration Important?

This consideration taken in mind the specific need of the developers to know which compiler/tool is the indicate to use in the construction of the software expected taking the security in the process of development as an important part to prevent the security bugs, flags, etc., that could emerge in the development process, this include the unsafe functions that the language in which the software is developed can have and ban them to prevent the security holes or problems that these make.

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

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 of 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

Component: Set of functional services in the software, which, when implemented, represents a well-defined set of functions and is distinguishable by a unique name [ISO/IEC 29881:2008]

Header: A header file is a file that contains C declarations and macro definitions (see Macros) that are shared across multiple source files. [GCC GNU]

Compiler: Software that translates a series of high-level language instructions into a low-level representation. For example, the Help compiler translates a text document containing the appropriate commands into an online help system. The pre-compiler translates terms and definitions in a dictionary search system [PCMAG]

Flags: Computer flags are values that act as signals for functions and processes. The value of the flag is used to determine the next step in the program. The flag is usually a binary flag containing a Boolean value (true or false). However, not all flags are binary, so you can store a range of values. [TechTerms]

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

3. Relationships with ISO/IEC 29110

This deployment package covers the activities related to Construction and Unit Test of the ISO Technical Report ISO/IEC 29110 Part 5-1-2 for Very Small Entities (VSEs) – Basic Profile [ISO/IEC29110].

The construction activities should have been planned during the Project planning activity of the project. The construction activities should be described in the project plan. If this is not the case, the project manager should perform this activity before beginning construction. (see the Project Management Deployment Package)

In this section, the reader will find a list of Project Management (PM) process, activities, tasks and roles from Part 5 that are directly related to this topic. This topic is described in details in the next section.

- **Process:** PM – Project Management
  - **Activity:** PM.1[1] Software Requirement Analysis
    - **Tasks and Roles:**

Tasks	Roles[2]
PM.1.5. Identify and document the Resources to perform the project both in development and security: human, material, equipment and tools, standards, including the required training of the Work Team, List of assets. Include in the schedule the dates when Resources and training will be needed.	PM, TL

[1] These numbers refer to processes, activities, tasks of ISO/IEC 29110 Part 5-1-2

[2] Roles are defined in a next section. Roles are also defined in ISO/IEC 29110 Part 5-1-2

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

4. Description of Processes, Activities, Tasks, Steps, Roles and Products

- **Process:** PM – Project Management
  - **Activity:** PM.1[3] Software Requirement Analysis
    - **Tasks and Roles:**

Tasks	Roles[4]
PM.1.5. Identify and document the Resources to perform the project both in development and security: human, material, equipment and tools, standards, including the required training of the Work Team, List of assets. Include in the schedule the dates when Resources and training will be needed.	PM, TL

This task is related with the following sub-tasks:

- Specify/Approve Secure Compilers, Tools, Flags & Options
- Identify/Deprecate Unsafe Functions

Sub-task: Specify/Approve Secure Compilers, Tools, Flags & Options

Objectives:	Define and publish a list of approved tools and their associated security checks, such as compiler/linker options and warnings
Rationale:	This list must be approved by the project team's security advisor In general, development teams should work to take advantage of new protections and security scans with the latest approved versions of the tools.
Roles:	Project Manager
	Technical Leader
	Programmer
	Security Advisor
Artefacts:	List of Secure Compilers, Tools, Flags & Options
Steps:	1. Identify Secure Compilers, Tools, Flags & Options.
	2. Enlist the Secure Compilers, Tools, Flags & Options required.
	3. Select the Secure Compilers, Tools, Flags & Options.
	4. Adopt the Secure Compilers, Tools, Flags & Options.
	5. Verify the adoption of the Secure Compilers, Tools, Flags &

[3] These numbers refer to processes, activities, tasks of ISO/IEC 29110 Part 5-1-2

[4] Roles are defined in a next section. Roles are also defined in ISO/IEC 29110 Part 5-1-2

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

	Options.
Step Description:	***Step 1. Identify Secure Compilers, Tools, Flags & Options*** Before the development phase, the Security Advisor must help in the definition of the development environment in which the development team must work. ***Step 2. Enlist the Secure Compilers, Tools, Flags & Options.*** With the help of the Security Advisor, enlist the options of Secure compilers, tools, flags & option to give to the development team. ***Step 3. Select the Secure Compilers, Tools, Flags & Options*** With the help of the Security Advisor, select the compiler, tool, flag or option that helps more in the secure development process. After selecting it, the Security Advisor must define a development environment for the development team to implement and work in it. ***Step 4. Adopt the Secure Compilers, Tools, Flags & Options.*** The development team must adopt the development environment in the phase of construction and report any issue regarding the defined environment. ***Step 5. Verify the adoption of the Secure Compilers, Tools, Flags & Options.*** Verify if the development team is adapting to the environment designed to use the programming language in the safest way.

Sub-task: Identify/Deprecate Unsafe Functions

Objectives:	Analyzes all features and APIs used with software development projects and prohibits them from being identified as insecure.
Rationale:	The project team should analyze all the features and APIs used in the software development project and ban them from what is considered dangerous. After defining the ban list, the project team can use header files (such as Bann.h or strsafe.h), the latest compilers, or code analysis tools to code for banned features (legacy code, if any). (Including it) needs to be verified. Replace prohibited features with safer alternatives.
Roles:	Technical Leader
	Programmer

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

Artefacts:	Banned list
Steps: ***(Programmer)***	1. Identify unsafe functions of the development language
	2. Make a list with all the identified functions
	3. Create Headers that contains the Banned functions
	4. Use the Headers in every file
Steps: ***(Technical Leader)***	1. Verify the Banned list
	2. Write a report with the Banned list
	3. Report the Banned list and the Headers created
Step Description: ***(Programmer)***	***Step 1.*** **Identify unsafe functions of the development language** Analyze all functions and APIs that will be used in conjunction with a software development project. ***Step 2.*** **Make a list with all the identified functions** Write a brief description of the unsafe functions including the reason of the classification as such. ***Step 3.*** **Create Headers that contains the Banned functions** Write headers or classes that include the banned functions to have it available to the development team. ***Step 4.*** **4. Use the Headers in every file** Include the Headers on every development file to ensure the secure use of the programming language.
Step Description: ***(Technical Leader)***	***Step 1.*** **Verify the Banned list** Verify the report made by the programmer. If there is some inconsistency ask the programmer to clarify the details. ***Step 2.*** **Write a report with the Banned list** Report the unsafe functions that were selected as such and added to the ban list, and the Headers made with them. ***Step 3.*** **Report the Banned list and the Headers created** Once the report is complete send it to the work team and if possible, make the list known by all the work team

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

Role Description

This is an alphabetical list of the roles, abbreviations and list of competencies as defined in ISO 29110 Part 5-1-2 and the security considerations.

	Role	*Abbreviation*	*Competency*
2.	Programmer	PR	Knowledge and/or experience in programming, integration and unit tests. Knowledge of the revision techniques. Knowledge of the editing techniques. Experience on the software development and maintenance.
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 process domain.
5.	Project Manager	PM	Leadership capability with experience making decisions, planning, personnel management, delegation and supervision, finances and software development.

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.

	Name	Description	Source

1.	Project Plan	Presents how the project processes and activities will be executed to assure the project's successful completion, and the quality of the deliverable products. It Includes the following elements which may have the characteristics as follows:
		- *Product Description*
		  ○ Purpose
		  ○ General Customer requirements
		- *Scope* description of what is included and what is not
		- *Objectives* of the project
		- *Deliverables* - list of products to be delivered to Customer
		- *Tasks, including* verification, validation and reviews with Customer and Work Team, to assure the quality of work products. Tasks may be represented as a Work Breakdown Structure (WBS). The task also includes the identification of security requirements, list of assets, threats, information security risk and
			Project Management

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

		incidents. - *Relationship and Dependence of the Tasks* - *Estimated Duration* of tasks - *Resources* o perform the project both in development and security (humans, materials, equipment and tools) including the required training, and the schedule when the resources are needed. - *Composition of Work Team* - *Competences pf personal Record* - *Role Matrix* - *Incidents Response of the project* - *Schedule of the Project Tasks,* the expected start and completion date, for each task. - *Estimated Effort and Cost* - *Identification of Project Risks* - *Version Control Strategy* - Product repository tools or mechanism identified - Location and access mechanisms for the repository specified - Version identification and control defined - Backup and recovery mechanisms defined - Storage, handling and delivery (including archival and retrieval) mechanisms specified - *Delivery Instructions* - Elements required for product release identified (i.e., hardware, software, documentation etc.) - Delivery requirements - Sequential ordering of tasks to be performed - Applicable releases identified - Identifies all delivered software components with version information - Identifies any necessary backup and recovery procedures The applicable statuses are: verified, validated, changed and reviewed.	
2.	*Software Component*	A set of related code units. The applicable statuses are: unit tested, corrected and baselined.	Software Implementation
3.	Software Design	This document includes textual and graphical information on the software structure. This structure may includes the following parts: Architectural High Level Software Design – Describes the overall *Software* structure: - Identifies the required software *Components* - Identifies the relationship between software *Components* - Consideration is given to any required:	Software Implementation

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

		- software performance characteristics - hardware, software and human interfaces - security characteristics - database design requirements - error handling and recovery attributes Detailed Low Level Software Design – includes details of the Software Components to facilitate its construction and test 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 data 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.	Traceability Record	Documents the relationship among the requirements included in the *Requirements Specification, Software Design* elements*, Software Components, Test Cases and Test Procedures. It may include:* - Identifies requirements of *Requirements Specification* to be traced - Provides forward and backwards mapping of requirements to *Software Design* elements, *Software Components*, *Test Cases and Test Procedures*. The applicable statuses are: verified, baselined and updated	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.	List of Secure Compilers, Tools, Flags	All development teams must define and publish a list of approved tools and compiler/linker options and associated security controls, including warnings. This list must be approved by the project team's security manager. In general, development teams need to work to deliver new

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

	Name	Description
	& Options	security scanning protections and features using the latest versions of approved tools.
2.	Banned List	This list is defined in the header with dangerous features identified in the language environment and should be added to each file used during the development phase of the project.

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

5. Template

5.1 List of Banned Functions & Safe String Alternatives in C language[1]

Banned Function	*Replacement Function*
alloca() **_alloca()**	use malloc() or new() which create memory on the heap, instead of the alloc functions which allocate memory on the stack, as alloc can allow damage to stack frames
scanf() **wscanf()** **sscanf()** **swscanf()** **vscanf()** **vsscanf()**	use fgets() instead of scanf() functions
strlen() **wcslen()**	strnlen_s() wcsnlen_s()
strtok() **strtok_r()** **wcstok()**	strtok_s()
strcat() **strncat()** **wcscat()** **wcsncat()**	strcat_s(), strncat_s(), **strlcat()*** wcscat_s(), wcsncat_s()
strcpy() **strncpy()** **wcscpy()** **wcsncpy()**	strcpy_s() strncpy_s(), **strlcpy()*** wcscpy_s(), wcsncpy_s()
memcpy() **wmemcpy()**	memcpy_s() wmemcpy_s()
stpcpy() **stpncpy()** **wcpcpy()** **wcpncpy()**	stpcpy_s(), stpncpy_s() wcpcpy_s(), wcpncpy_s()
memmove() **wmemmove()**	memmove_s() wmemmove_s()

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

memcmp() **wmemcmp()**	memcmp_s() wmemcmp_s()
memset() **wmemset()**	memset_s() wmemset_s()
gets()	use fgets() instead
sprintf() **vsprintf()** **swprintf()** **vswprintf()**	use snprintf() or one of the specialized (non-varg) versions in the safe string library
snprintf() **vsnprintf()**	Consider using a wrapper function that avoids the vargs construct and uses compile-time checks on the parameters passed into snprintf(). See example functions in the Safe String library.
realpath()	continue to use realpath() but use NULL for the second parameter to force allocation of an appropriate sized buffer on the heap.
getwd()	use getcwd() instead because it checks the buffer size
wctomb() **wcrtomb()** **wcstombs()** **wcsrtombs()** **wcsnrtombs()**	The wide-character to multi-byte string conversion routines can create buffer overflows, but currently no alternatives are provided. If enough requests are made that indicate these functions are in wide use and safer alternatives are needed, these functions may be added to the library extensions.

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

6. References

Key	Reference
[Code Complete]	Steve McConnell, Code Complete, Second Edition, Redmond, Washington, Microsoft Press, 2004**.**
[Art of Software testing]	Glenford J. Myers, The Art of Software Testing, Second Edition, 2004.
[Practitioner's Guide]	Lee Copeland, A Practitioner's Guide to Software Test Design, 2004
[Defect Prevention]	Marc McDonald, The Practical Guide To Defect Prevention, 2008
[Introduction to Software Testing]	Paul Ammann & Jeff Offutt, Introduction to Software testing, 2008
[Testing Computer Software]	Cem Kaner, Testing Computer Software
[Pratical Software Testing]	Ilene Burnstein, Practical Software Testing, 2002
[SE Support Activities for VSE]	Vincent Ribaud, Software Engineering Support Activities for Very Small Entities, 2010
[Application of ISES in VSE]	Claude Y. Laporte, The application of International Software Engineering Standards in Very Small Enterprises, 2008
[A SE Lifecycle Standard for VSEs]	Claude Y. Laporte, A Software Engineering Lifecycle Standard for Very Small Enterprises, 2008
[Misuse Code Coverage]	Brian Marick, How to Misuse Code Coverage, 1999
[IEEE 1012-2004]	IEEE 1012-2004 IEEE Standard for Software Verification and Validation, IEEE Computer Society
[ISO/IEC 12207]	ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.
[ISO/IEC TR 29110-5-1-2]	ISO/IEC TR 29110-5-1-2:2011, Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs) – Part 5-1-2: Management and Engineering Guide – Generic Profile Group - Basic Profile
[ISO/IEC 24765]	ISO/IEC 24765:2010 Systems and software engineering vocabulary
[ISO/IEC 20926]	ISO/IEC 20926:2003 Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting practices manual
[ISO/IEC 29881:2008]	ISO/IEC 29881:2008 Information technology--Software and systems engineering--FiSMA 1.1 functional size measurement method,
[IEEE 1233-1998]	IEEE Guide for Developing System Requirements Specifications

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives

[PCMAG]	PC MAG Online: https://www.pcmag.com/encyclopedia/term/compiler
[GCC GNU]	Gcc, gnu, Online: https://gcc.gnu.org/onlinedocs/cpp/Header-Files.html
[TechTerms]	TechTerms, Flag, Online: https://techterms.com/definition/flag

1 https://github.com/intel/safestringlib/wiki/SDL-List-of-Banned-Functions#list-of-banned-functions--safe-string-alternatives