IEC 81001-5-1

Edition 1.0 2021-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Health software and health IT systems safety, effectiveness and security – Part 5-1: Security – Activities in the product life cycle

Logiciels de santé et sécurité, efficacité et sûreté des systèmes TI de santé – Partie 5-1: Sûreté – Activités du cycle de vie du produit

IEC 81001-5-1:2021-12(en-fr)

IEC 81001-5-1

Edition 1.0 2021-12

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Health software and health IT systems safety, effectiveness and security – Part 5-1: Security – Activities in the product life cycle

Logiciels de santé et sécurité, efficacité et sûreté des systèmes TI de santé – Partie 5-1: Sûreté – Activités du cycle de vie du produit

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 11.040.01; 35.240.80

ISBN 978-2-8322-5578-0

CONTENTS

FOREWORD ... 5
INTRODUCTION ... 7
0.1 Structure ... 7
0.2 Field of application ... 8
0.3 Conformance ... 8
1 Scope ... 10
2 Normative references ... 10
3 Terms and definitions ... 11
4 General requirements ... 18
4.1 Quality management ... 18
4.1.1 Quality management system ... 18
4.1.2 Identification of responsibilities ... 18
4.1.3 Identification of applicability ... 18
4.1.4 SECURITY expertise ... 18
4.1.5 SOFTWARE ITEMS from third-party suppliers ... 19
4.1.6 Continuous improvement ... 19
4.1.7 Disclosing SECURITY-related issues ... 19
4.1.8 Periodic review of SECURITY defect management ... 19
4.1.9 ACCOMPANYING DOCUMENTATION review ... 20
4.2 SECURITY RISK MANAGEMENT ... 20
4.3 SOFTWARE ITEM classification relating to risk transfer ... 20
5 Software development PROCESS ... 21
5.1 Software development planning ... 21
5.1.1 ACTIVITIES in the LIFE CYCLE PROCESS ... 21
5.1.2 Development environment SECURITY ... 21
5.1.3 Secure coding standards ... 21
5.2 HEALTH SOFTWARE requirements analysis ... 21
5.2.1 HEALTH SOFTWARE SECURITY requirements ... 21
5.2.2 SECURITY requirements review ... 22
5.2.3 SECURITY risks for REQUIRED SOFTWARE ... 22
5.3 Software architectural design ... 22
5.3.1 DEFENSE-IN-DEPTH ARCHITECTURE/design ... 22
5.3.2 Secure design best practices ... 22
5.3.3 SECURITY architectural design review ... 23
5.4 Software design ... 23
5.4.1 Software design best practices ... 23
5.4.2 Secure design ... 23
5.4.3 Secure HEALTH SOFTWARE interfaces ... 23
5.4.4 Detailed design VERIFICATION for SECURITY ... 24
5.5 Software unit implementation and VERIFICATION ... 24
5.5.1 Secure coding standards ... 24
5.5.2 SECURITY implementation review ... 24
5.6 Software integration testing ... 25
5.7 Software system testing ... 25
5.7.1 SECURITY requirements testing ... 25
5.7.2 THREAT mitigation testing ... 25

5.7.3 VULNERABILITY testing 25
5.7.4 Penetration testing 26
5.7.5 Managing conflicts of interest between testers and developers 26
5.8 Software release 26
5.8.1 Resolve findings prior to release 26
5.8.2 Release documentation 27
5.8.3 File INTEGRITY 27
5.8.4 Controls for private keys 27
5.8.5 Assessing and addressing SECURITY-related issues 27
5.8.6 ACTIVITY completion 27
5.8.7 SECURE decommissioning guidelines for HEALTH SOFTWARE 27
6 SOFTWARE MAINTENANCE PROCESS 28
6.1 Establish SOFTWARE MAINTENANCE plan 28
6.1.1 Timely delivery of SECURITY updates 28
6.2 Problem and modification analysis 28
6.2.1 Monitoring public incident reports 28
6.2.2 SECURITY update VERIFICATION 28
6.3 Modification implementation 29
6.3.1 SUPPORTED SOFTWARE SECURITY update documentation 29
6.3.2 MAINTAINED SOFTWARE SECURITY update delivery 29
6.3.3 MAINTAINED SOFTWARE SECURITY update INTEGRITY 29
7 SECURITY RISK MANAGEMENT PROCESS 29
7.1 RISK MANAGEMENT context 29
7.1.1 General 29
7.1.2 PRODUCT SECURITY CONTEXT 29
7.2 Identification of VULNERABILITIES, THREATS and associated adverse impacts 30
7.3 Estimation and evaluation of SECURITY risk 31
7.4 Controlling SECURITY risks 31
7.5 Monitoring the effectiveness of RISK CONTROLS 31
8 Software CONFIGURATION MANAGEMENT PROCESS 32
9 Software problem resolution PROCESS 32
9.1 Overview 32
9.2 Receiving notifications about VULNERABILITIES 32
9.3 Reviewing VULNERABILITIES 32
9.4 Analysing VULNERABILITIES 33
9.5 Addressing SECURITY-related issues 33
Annex A (informative) Rationale 35
A.1 Relationship to IEC 62443 35
A.2 Relationship to IEC 62304 36
A.3 Risk transfer 37
A.3.1 Overview 37
A.3.2 MAINTAINED SOFTWARE 37
A.3.3 SUPPORTED SOFTWARE 37
A.3.4 REQUIRED SOFTWARE 37
A.4 Secure coding best practices 38
Annex B (informative) Guidance on implementation of SECURITY LIFE CYCLE ACTIVITIES 39
B.1 Overview 39
B.2 Related work 39

B.3 THREAT / RISK ANALYSIS 39
B.4 THREAT and RISK MANAGEMENT 40
B.5 Software development planning 40
B.5.1 Development 40
B.5.2 HEALTH SOFTWARE requirements analysis 41
B.5.3 Software architectural design 41
B.5.4 Software unit implementation and VERIFICATION 41
B.5.5 Secure implementation 42
B.5.6 Not used 42
B.5.7 Software system testing 42
Annex C (informative) THREAT MODELLING 44
C.1 General 44
C.2 ATTACK-defense trees 44
C.3 CAPEC / OWASP / SANS 44
C.4 CWSS 44
C.5 DREAD 45
C.6 List known potential VULNERABILITIES 45
C.7 OCTAVE 45
C.8 STRIDE 45
C.9 Trike 45
C.10 VAST 45
Annex D (informative) Relation to practices in IEC 62443-4-1:2018 46
D.1 IEC 81001-5-1 to IEC 62443-4-1:2018 46
D.2 IEC 62443-4-1:2018 to IEC 81001-5-1 47
Annex E (informative) Documents specified in IEC 62443-4-1 48
E.1 Overview 48
E.2 Release documentation 48
E.2.1 PRODUCT documentation 48
E.2.2 HEALTH SOFTWARE DEFENSE-IN-DEPTH documentation 49
E.2.3 DEFENSE-IN-DEPTH measures expected in the environment 49
E.2.4 SECURITY hardening guidelines 49
E.2.5 SECURITY update information 50
E.3 Documents for decommissioning HEALTH SOFTWARE 50
Annex F (normative) TRANSITIONAL HEALTH SOFTWARE 51
F.1 Overview 51
F.2 Development assessment and gap closure activities 51
F.3 Rationale for use of TRANSITIONAL HEALTH SOFTWARE 52
F.4 Post-release ACTIVITIES 52
Annex G (normative) Object identifiers 53
Bibliography 54

Figure 1 – HEALTH SOFTWARE field of application 8
Figure 2 – HEALTH SOFTWARE LIFE CYCLE PROCESSES 10

Table A.1 – Required level of independence of testers from developers 36
Table G.1 – Object identifiers for conformance concepts of this document 53

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

HEALTH SOFTWARE AND HEALTH IT SYSTEMS SAFETY, EFFECTIVENESS AND SECURITY –

Part 5-1: Security – Activities in the product life cycle

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 81001-5-1 has been prepared by a Joint Working Group of IEC subcommittee 62A: Common aspects of electrical equipment used in medical practice, of IEC technical committee 62: Electrical equipment in medical practice, and ISO technical committee 215: Health informatics.

It is published as a double logo standard.

The text of this document is based on the following documents:

Draft	Report on voting
62A/1458/FDIS	62A/1466/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

In this document, the following print types are used:

– requirements and definitions: roman type;

– informative material appearing outside of tables, such as notes, examples and references: in smaller type. Normative text of tables is also in a smaller type;

– TERMS DEFINED IN CLAUSE 3 OF THE GENERAL STANDARD, IN THIS PARTICULAR STANDARD OR AS NOTED: SMALL CAPITALS.

A list of all parts in the IEC 81001 series, published under the general title *Health software and health IT systems safety, effectiveness and security*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

0.1 Structure

PROCESS standards for HEALTH SOFTWARE provide a specification of ACTIVITIES that will be performed by the MANUFACTURER – including software incorporated in medical devices –as a part of a development LIFE CYCLE. The normative clauses of this document are intended to provide minimum best practices for a secure software LIFE CYCLE. Local legislation and regulation are considered.

PROCESS requirements (Clause 4 through Clause 9) have been derived from the IEC 62443-4-1[11][1] PRODUCT LIFE CYCLE management. Implementations of these specifications can extend existing PROCESSES at the MANUFACTURER's organization – notably existing PROCESSES conforming to IEC 62304[8]. This document can therefore support conformance to IEC 62443-4-1[11].

Normative clauses of this document specify ACTIVITIES that are the responsibility of the MANUFACTURER. The HEALTH SOFTWARE LIFE CYCLE can be part of an incorporating PRODUCT project. Some ACTIVITIES specified in this document depend on input and support from the PRODUCT LIFE CYCLE (for example to define specific criteria). Examples include:

- RISK MANAGEMENT;
- requirements;
- testing;
- post-release (after first placing HEALTH SOFTWARE on the market).

In cases where ACTIVITIES for HEALTH SOFTWARE need support from PROCESSES at the PRODUCT level, Clause 4 through Clause 9 of this document specify respective requirements beyond the HEALTH SOFTWARE LIFE CYCLE.

Similar to IEC 62304[8], this document does not prescribe a specific system of PROCESSES, but Clause 4 through Clause 9 of this document specify ACTIVITIES that are performed during the HEALTH SOFTWARE LIFE CYCLE.

Clause 4 specifies that MANUFACTURERS develop and maintain HEALTH SOFTWARE within a quality management system (see 4.1) and a RISK MANAGEMENT SYSTEM (4.2).

Clause 5 through Clause 8 specify ACTIVITIES and resulting output as part of the software LIFE CYCLE PROCESS implemented by the MANUFACTURER. These specifications are arranged in the ordering of IEC 62304[8].

Clause 9 specifies ACTIVITIES and resulting output as part of the problem resolution PROCESS implemented by the MANUFACTURER.

The scope of this document is limited to HEALTH SOFTWARE and its connectivity to its INTENDED ENVIRONMENT OF USE, based on IEC 62304[8], but with emphasis on CYBERSECURITY.

For expression of provisions in this document,

– "can" is used to describe a possibility or capability; and
– "must" is used to express an external constraint.

[1] Numbers in square brackets refer to the Bibliography.

NOTE HEALTH SOFTWARE can be placed on the market as software, as part of a medical device, as part of hardware specifically intended for health use, as a medical device (SaMD), or as a PRODUCT for other health use. (See Figure 2).

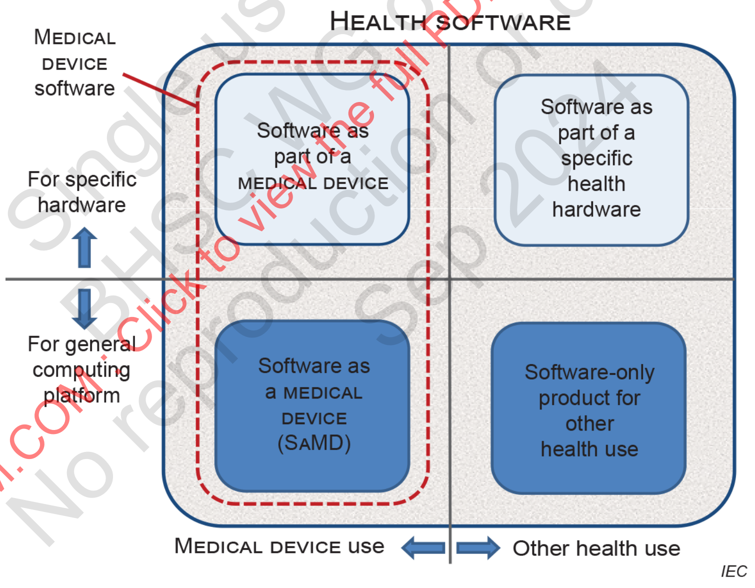
0.2 Field of application

This document applies to the development and maintenance of HEALTH SOFTWARE by a MANUFACTURER, but recognizes the critical importance of bi-lateral communication with organizations (e.g. HEALTHCARE DELIVERY ORGANIZATIONS, HDOs) who have SECURITY responsibilities for the HEALTH SOFTWARE and the systems it is incorporated into, once the software has been developed and released. The ISO/IEC 81001-5 series of standards (for which this is part -1), is therefore being designed to include future parts addressing SECURITY that apply to the implementation, operations and use phases of the LIFE CYCLE for organizations such as HDOs.

A medical device software is a subset of HEALTH SOFTWARE. A practical Venn diagram of HEALTH SOFTWARE types is shown in Figure 1. Therefore, this document applies to:

– software as part of a medical device;
– software as part of hardware specifically intended for health use;
– software as a medical device (SaMD); and
– software-only PRODUCT for other health use.

NOTE In this document, the scope of software considered part of the LIFE CYCLE ACTIVITIES for secure HEALTH SOFTWARE is larger and includes more software (drivers, platforms, operating systems) than for SAFETY, because for SECURITY the focus will be on any use including foreseeable unauthorized access rather than just the INTENDED USE.


HEALTH SOFTWARE
MEDICAL DEVICE software
For specific hardware
For general computing platform
Software as part of a MEDICAL DEVICE
Software as part of a specific health hardware
Software as a MEDICAL DEVICE (SaMD)
Software-only product for other health use
MEDICAL DEVICE use
Other health use
IEC

[SOURCE: IEC 82304-1[18]]

Figure 1 – HEALTH SOFTWARE field of application

0.3 Conformance

Conformance with this document focuses on the implementation of requirements regarding PROCESSES, ACTIVITIES, and TASKS – and can be claimed in one of two alternative ways:

- for HEALTH SOFTWARE by implementing requirements in Clause 4 through Clause 9 of this document,
- for TRANSITIONAL HEALTH SOFTWARE by only implementing the PROCESSES, ACTIVITIES, and TASKS identified in Annex F.

This document is designed to assist in the implementation of the PROCESSES required by IEC 62443-4-1, however, conformance to this document is not necessarily a sufficient condition for conformance to IEC 62443-4-1[11]. More guidance on coverage can be found in Annex D.

MANUFACTURERS can implement the specifications for Annex E in order to achieve conformance of documentation to IEC 62443-4-1[11].

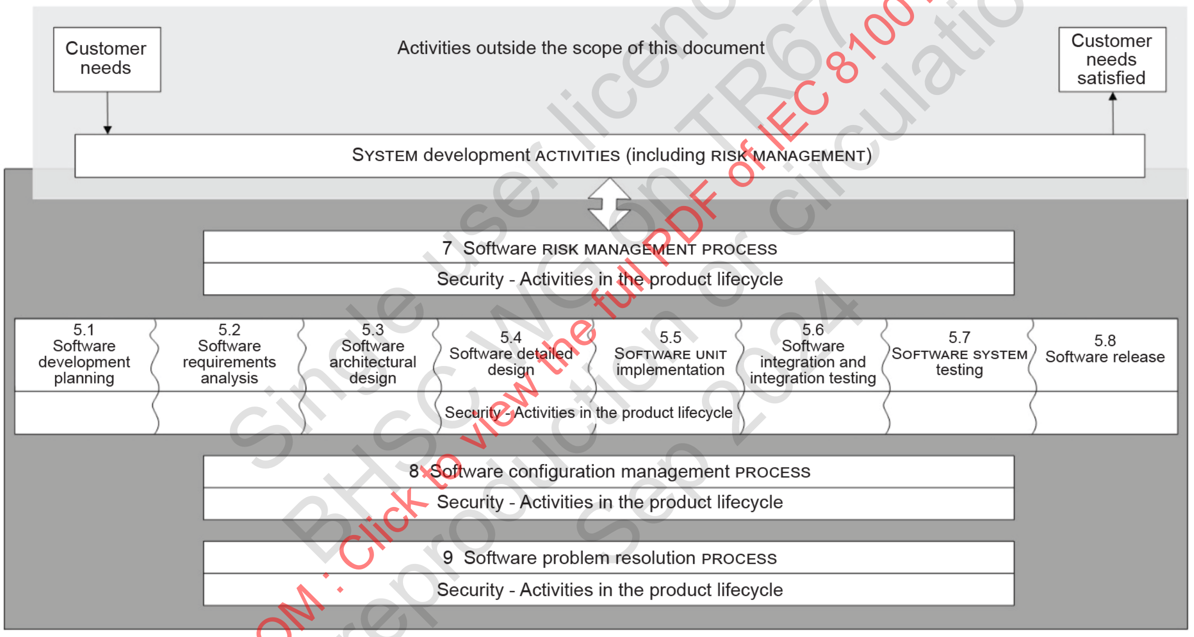
Clause 4 through Clause 9 of this document require establishing one or more PROCESSES that include identified ACTIVITIES. Per these normative parts of this document, the LIFE CYCLE PROCESSES implement these ACTIVITIES. None of the requirements in this document requires to implement these ACTIVITIES as one single PROCESS or as separate PROCESSES. The ACTIVITIES specified in this document will typically be part of an existing LIFE CYCLE PROCESS.

HEALTH SOFTWARE AND HEALTH IT SYSTEMS SAFETY, EFFECTIVENESS AND SECURITY –

Part 5-1: Security – Activities in the product life cycle

1 Scope

This document defines the LIFE CYCLE requirements for development and maintenance of HEALTH SOFTWARE needed to support conformance to IEC 62443-4-1[11] – taking the specific needs for HEALTH SOFTWARE into account. The set of PROCESSES, ACTIVITIES, and TASKS described in this document establishes a common framework for secure HEALTH SOFTWARE LIFE CYCLE PROCESSES. An informal overview of activities for HEALTH SOFTWARE is shown in Figure 2.


Customer needs
Activities outside the scope of this document
Customer needs satisfied
SYSTEM development ACTIVITIES (including RISK MANAGEMENT)
7 Software RISK MANAGEMENT PROCESS
Security - Activities in the product lifecycle
5.1 Software development planning
5.2 Software requirements analysis
5.3 Software architectural design
5.4 Software detailed design
5.5 SOFTWARE UNIT implementation
5.6 Software integration and integration testing
5.7 SOFTWARE SYSTEM testing
5.8 Software release
Security - Activities in the product lifecycle
8 Software configuration management PROCESS
Security - Activities in the product lifecycle
9 Software problem resolution PROCESS
Security - Activities in the product lifecycle
IEC

[derived from IEC 62304:2006[8], Figure 2]

Figure 2 – HEALTH SOFTWARE LIFE CYCLE PROCESSES

The purpose is to increase the CYBERSECURITY of HEALTH SOFTWARE by establishing certain ACTIVITIES and TASKS in the HEALTH SOFTWARE LIFE CYCLE PROCESSES and also by increasing the SECURITY of SOFTWARE LIFE CYCLE PROCESSES themselves.

It is important to maintain an appropriate balance of the key properties SAFETY, effectiveness and SECURITY as discussed in ISO 81001-1[17].

This document excludes specification of ACCOMPANYING DOCUMENTATION contents.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at www.electropedia.org/
- ISO Online browsing platform: available at www.iso.org/obp

3.1
ACCOMPANYING DOCUMENTATION
documentation intended to be used for a HEALTH SOFTWARE or a HEALTH IT SYSTEM or an accessory, containing information for the responsible organization or operator

3.2
ACTIVITY
set of one or more interrelated or interacting TASKS

[SOURCE: IEC 62304:2006[8], 3.1]

3.3
ARCHITECTURE
fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution

[SOURCE: ISO/IEC/IEEE 24765:2017, 3.216, definition1]

3.4
ASSET
physical or digital entity that has value to an individual, an organization or a government

Note 1 to entry: As per the definition for ASSET this can include the following:

a) data and information;
b) HEALTH SOFTWARE and software needed for its operation;
c) hardware components such as computers, mobile devices, servers, databases, and networks;
d) services, including SECURITY, software development, IT operations and externally provided services such as data centres, internet and software-as-a-service and cloud solutions;
e) people, and their qualifications, skills and experience;
f) technical procedures and documentation to manage and support the HEALTH IT INFRASTRUCTURE;
g) HEALTH IT SYSTEMS that are configured and implemented to address organizational objectives by leveraging the ASSETS; and
h) intangibles, such as reputation and image.

[SOURCE: ISO 81001-1:2021[17] 3.3.2, modified – Addition of a new Note 1 to entry.]

3.5
ATTACK
attempt to destroy, expose, alter, disable, steal or gain unauthorized access to or make unauthorized use of an ASSET

[SOURCE: ISO/IEC 27000:2018, 3.2]

3.6
ATTACK SURFACE
physical and functional interfaces of a system that can be accessed and, therefore, potentially exploited by an attacker

[SOURCE: IEC 62443-4-1:2018[11], 3.1.7]

3.7
AVAILABILITY
property of being accessible and usable on demand by an authorized entity

[SOURCE: ISO/IEC 27000:2018, 3.7]

3.8
CONFIDENTIALITY
property that information is not made available or disclosed to unauthorized individuals, entities, or PROCESSES

[SOURCE: ISO/IEC 27000:2018, 3.10]

3.9
CONFIGURATION ITEM
entity that can be uniquely identified at a given reference point

[SOURCE: IEC 62304:2006[8], 3.5]

3.10
CONFIGURATION MANAGEMENT
PROCESS ensuring consistency of CONFIGURATION ITEMS by using mechanisms for identifying, controlling and tracking versions of CONFIGURATION ITEMS

3.11
DEFENSE-IN-DEPTH
approach to defend the system against any particular ATTACK using several independent methods

Note 1 to entry: DEFENSE-IN-DEPTH implies layers of SECURITY and detection, even on single systems, and provides the following features:

- is based on the idea that any one layer of protection, can and probably will be defeated;
- attackers are faced with breaking through or bypassing each layer without being detected;
- a flaw in one layer can be mitigated by capabilities in other layers;
- system SECURITY becomes a set of layers within the overall network SECURITY; and
- each layer is autonomous and not rely on the same functionality nor have the same failure modes as the other layers.

[SOURCE: IEC 62443-4-1:2018[11], 3.1.15]

3.12
EXPLOIT (noun)
defined way to breach the SECURITY of information systems through some VULNERABILITY

[SOURCE: ISO/IEC 27039:2015, 2.9]

3.13
HEALTH IT INFRASTRUCTURE
combined set of IT ASSETS available to the individual or organization for developing, configuring, integrating, maintaining, and using IT services and supporting health, patient care and other organizational objectives

[SOURCE: ISO 81001-1:2021[17], 3.3.7, modified – Deletion of the Note 1 to entry.]

3.14
HEALTH IT SYSTEM
a combination of interacting health information elements (including HEALTH SOFTWARE, medical devices, IT hardware, interfaces, data, procedures and documentation) that is configured and implemented to support and enable an individual or organization's specific health objectives

[SOURCE: ISO 81001-1:2021[17], 3.3.8, modified – Addition of "(including HEALTH SOFTWARE, medical devices, IT hardware, interfaces, data, procedures and documentation)".]

3.15
HEALTH SOFTWARE
software intended to be used specifically for managing, maintaining, or improving health of individual persons, or the delivery of care, or which has been developed for the purpose of being incorporated into a medical device

Note 1 to entry: HEALTH SOFTWARE fully includes what is considered software as a medical device.

[SOURCE: ISO 81001-1:2021[17], 3.3.9]

3.16
HEALTHCARE DELIVERY ORGANIZATION
HDO
facility or enterprise such as a clinic or hospital that provides healthcare services

[SOURCE: ISO 81001-1:2021[17], 3.1.4]

3.17
INTEGRITY
property of accuracy and completeness

[SOURCE: ISO/IEC 27000:2018, 3.36]

3.18
INTENDED ENVIRONMENT OF USE
conditions and setting in which users interact with the HEALTH SOFTWARE – as specified by the MANUFACTURER

3.19
INTENDED USE
INTENDED PURPOSE
use for which a PRODUCT, PROCESS or service is intended according to the specifications, instructions and information provided by the MANUFACTURER

Note 1 to entry: The intended medical indication, patient population, part of the body or type of tissue interacted with, user profile, INTENDED ENVIRONMENT OF USE, and operating principle are typical elements of the INTENDED USE.

[SOURCE: ISO 81001-1:2021[17], 3.2.7, modified – In Note 1 to entry, replacement of "USE ENVIRONMENT" with "INTENDED ENVIRONMENT OF USE".]

3.20
LIFE CYCLE
series of all phases in the life of a PRODUCT or system, from the initial conception to final decommissioning and disposal

[SOURCE: ISO 81001-1:2021[17], 3.3.12]

3.21
MAINTAINED SOFTWARE
SOFTWARE ITEM for which the MANUFACTURER will assume the risk related to SECURITY

Note 1 to entry: See also A.3.

3.22
MANUFACTURER
organization with responsibility for design or manufacture of a PRODUCT

Note 1 to entry: Responsibility extends to supporting ACTIVITIES during operations.

Note 2 to entry: There is only one MANUFACTURER, but technical responsibility can be with multiple entities along the supply chain, with service providers, or with entities at different stages in the LIFE CYCLE.

Note 3 to entry: Independent of the MANUFACTURER's responsibility, any specific legal accountability is defined by contracts and legislation.

[SOURCE: ISO 81001-1:2021[17], 3.1.7– Addition of the notes to entry.]

3.23
PROCESS
set of interrelated or interacting ACTIVITIES that use inputs to deliver an intended result (outcome)

[SOURCE: ISO 81001-1:2021[17], 3.2.10, modified – Added "(outcome)" after "result".]

3.24
PRODUCT
output of an organization that can be produced without any transaction taking place between the organization and the customer

Note 1 to entry: Production of a PRODUCT is achieved without any transaction necessarily taking place between provider and customer, but can often involve this service element upon its delivery to the customer.

Note 2 to entry: The dominant element of a PRODUCT is that it is generally tangible.

[SOURCE: ISO 81001-1:2021[17], 3.3.15]

3.25
REQUIRED SOFTWARE
SOFTWARE ITEM for which the MANUFACTURER will consider SECURITY-related risks known before release of the HEALTH SOFTWARE

Note 1 to entry: This includes SUPPORTED SOFTWARE. See A.3.

3.26
RESIDUAL RISK
risk remaining after RISK CONTROL measures have been implemented

[SOURCE: ISO 81001-1:2021[17], 3.4.9]

3.27
RISK CONTROL
PROCESS in which decisions are made and measures implemented by which risks are reduced to, or maintained within, specified levels

[SOURCE: ISO 81001-1:2021[17], 3.4.13, modified – Replacement of "limits" with "levels".]

3.28
RISK MANAGEMENT
systematic application of management policies, procedures and practices to the TASKS of analysing, evaluating, controlling and monitoring risk

[SOURCE: ISO 81001-1:2021[17], 3.4.16]

3.29
SAFETY
freedom from unacceptable risk

Note 1 to entry: In the context of SAFETY, risk is the combination of probability of occurrence of harm and severity of harm (see ISO/IEC Guide 51:2014).

Note 2 to entry: SECURITY incidents can lead to harm and can therefore have an impact on SAFETY.

[SOURCE: ISO 81001-1:2021[17], 3.2.12, modified – Addition of the notes to entry.]

3.30
SECURITY
CYBERSECURITY
state where information and systems are protected from unauthorized ACTIVITIES, such as access, use, disclosure, disruption, modification, or destruction to a degree that the related risks to violation of CONFIDENTIALITY, INTEGRITY, and AVAILABILITY are maintained at an acceptable level throughout the LIFE CYCLE

[SOURCE: ISO 81001-1:2021[17], 3.2.13]

3.31
SECURITY CAPABILITY
broad category of technical, administrative or organizational controls to manage risks to CONFIDENTIALITY, INTEGRITY, AVAILABILITY and accountability of data and systems

[SOURCE: ISO 81001-1:2021[17], 3.2.14]

3.32
SECURITY CONTEXT
minimum requirements and assumptions about the environment of HEALTH SOFTWARE – derived from the INTENDED ENVIRONMENT OF USE at PRODUCT-level, considering also the configuration and integration of HEALTH SOFTWARE and taking into account foreseeable unauthorized or unintended access

3.33
SOFTWARE COMPOSITION ANALYSIS
(electronic) analysis of binaries

Note 1 to entry: SOFTWARE COMPOSITION ANALYSIS can be supported by tools or online services.

3.34
SOFTWARE ITEM
identifiable part of a computer program, i.e. source code, object code, control code, control data, or a collection of these items

[SOURCE: IEC 62304:2006 and IEC 62304:2006/AMD1:2015, 3.25, modified – Deletion of the note.]

3.35
SOFTWARE MAINTENANCE
modification of HEALTH SOFTWARE after release

Note 1 to entry: Maintenance keeps the INTENDED USE and is done for one or more of the following reasons:

a) corrective, as fixing faults;

b) adaptive, as adapting to new hardware or software platform;

c) perfective, as implementing new requirements;

d) preventive, as making the PRODUCT more maintainable.

Note 2 to entry: See also ISO/IEC 14764:2006, 3.10.

[SOURCE: IEC 82304-1:2016, 3.21, modified – In the definition, the words "HEALTH SOFTWARE PRODUCT" have been replaced by "HEALTH SOFTWARE"; purposes of maintenance have been placed into a note that also highlights keeping the INTENDED USE, reference 3.10 has been added to the second note to entry; and "hard-" has been replaced by "hardware".]

3.36
SUPPORTED SOFTWARE
SOFTWARE ITEM for which the MANUFACTURER will notify the customer regarding known risks related to SECURITY

Note 1 to entry: This includes MAINTAINED SOFTWARE. See A.3.

3.37
TASK
single piece of work that needs to be done to achieve a specific goal

[SOURCE: IEC 62304:2006[8], 3.31, modified – Addition of "to achieve a specific goal".]

3.38
THREAT
potential for violation of SECURITY, which exists when there is a circumstance, capability, action, or event that could breach SECURITY and cause damage to CONFIDENTIALITY, INTEGRITY, AVAILABILITY of information ASSETS

[SOURCE: ISO 81001-1:2021[17], 3.4.1.21, modified – "Harm" replaced with "damage to CONFIDENTIALITY, INTEGRITY, AVAILABILITY of information ASSETS".]

3.39
THREAT MODEL
documented result of the THREAT MODELLING ACTIVITY

3.40
THREAT MODELLING
systematic exploration technique to expose any circumstance or event having the potential to cause damage to a system in the form of destruction, disclosure, modification of data, or denial of service

[SOURCE. ISO/IEC/IEEE 24765:2017, 3.4290, modified – Replacement of "harm" with "damage".]

3.41
TRACEABILITY
link between the origin of requirements throughout the project LIFE CYCLE to design elements, and test cases

3.42
TRANSITIONAL HEALTH SOFTWARE
HEALTH SOFTWARE, which was released prior to publication of this document and which does not meet all requirements specified in Clause 4 through Clause 9 of this document

Note 1 to entry: For TRANSITIONAL HEALTH SOFTWARE its MANUFACTURER can claim conformance to the Annex F.

3.43
TRUST BOUNDARY
element of a THREAT MODEL that depicts a boundary where authentication is required or a change in trust level occurs (higher to lower or vice versa)

Note 1 to entry: TRUST BOUNDARY enforcement mechanisms for PRODUCT users typically include authentication (for example, challenge/response, passwords, biometrics or digital signatures) and associated authorization (for example, access control rules).

Note 2 to entry: TRUST BOUNDARY enforcement mechanisms for data typically include source authentication (for example, message authentication codes and digital signatures) and/or content VALIDATION.

3.44
USE ENVIRONMENT
actual conditions and setting in which users interact with the HEALTH SOFTWARE

Note 1 to entry: For the purpose of this document, that includes data interfaces.

[SOURCE: IEC 62366-1:2015 and IEC 62366-1:2015/AMD1:2020, 3.20, modified – "Medical device" replaced with "HEALTH SOFTWARE" in the definition, and replacement of Note 1 to entry.]

3.45
VALIDATION
confirmation, through the provision of objective evidence, that the requirements for a specific INTENDED USE or application have been fulfilled

Note 1 to entry: The objective evidence needed for a VALIDATION is the result of a test or other form of determination such as performing alternative calculations or reviewing documents.

Note 2 to entry: The word “validated” is used to designate the corresponding status.

Note 3 to entry: The use conditions for VALIDATION can be real or simulated.

[SOURCE: ISO 9000:2015, 3.8.13]

3.46
VERIFICATION
confirmation, through provision of objective evidence, that specified requirements have been fulfilled

Note 1 to entry: The objective evidence needed for a VERIFICATION can be the result of an inspection or of other forms of determination such as performing alternative calculations or reviewing documents.

Note 2 to entry: The ACTIVITIES carried out for VERIFICATION are sometimes called a qualification PROCESS.

Note 3 to entry: The word "verified" is used to designate the corresponding status.

[SOURCE: ISO 81001-1:2021[17], 3.2.16]

3.47
VULNERABILITY
flaw or WEAKNESS in a system's design, implementation, or operation and management that could be exploited to violate the system's SECURITY policy

[SOURCE: ISO 81001-1:2021[17], 3.4.22]

3.48
WEAKNESS
kind of deficiency

Note 1 to entry: A WEAKNESS can result in a SECURITY risk.

[SOURCE: ISO 81001-1:2021[17], 3.4.23, modified – Deletion of "and/or privacy risks" in Note 1 to entry.]

4 General requirements

4.1 Quality management

4.1.1 Quality management system

The MANUFACTURER shall perform SECURITY ACTIVITIES in the PRODUCT LIFE CYCLE on the basis of an established and documented quality management system.

The quality management system can be implemented according to ISO 13485 or other equivalent quality management system standards.

Throughout this document "establish an ACTIVITY (or ACTIVITIES)" means that the MANUFACTURER shall document this ACTIVITY (or ACTIVITIES) and shall ensure that this ACTIVITY (or ACTIVITIES) is done effectively and completely.

4.1.2 Identification of responsibilities

The MANUFACTURER shall designate and document the organizational roles and personnel responsible for each of the ACTIVITIES and PROCESSES required by this document.

NOTE Personnel can be identified through functional roles instead of names.

4.1.3 Identification of applicability

The MANUFACTURER shall identify the PRODUCTS or parts of PRODUCTS to which the secure LIFE CYCLE applies.

NOTE 1 For HEALTH SOFTWARE some IT exposure, networking, or data interfacing capabilities are assumed and therefore a secure software LIFE CYCLE is followed.

NOTE 2 This requirement is not about PRODUCT instances (and their identification) but about types of PRODUCTS or their parts – for example SOFTWARE ITEMS. Having this ACTIVITY (or ACTIVITIES) means that the MANUFACTURER has criteria for identifying which parts of its PRODUCTS are developed, maintained and supported using the ACTIVITIES described by this document.

4.1.4 SECURITY expertise

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for identifying and providing SECURITY training and assessment programs to ensure that personnel assigned to the organizational roles and duties specified in 4.1.2 have demonstrated SECURITY expertise appropriate for those PROCESSES. Results of this ACTIVITY (or ACTIVITIES) include role descriptions, training profiles and training records.

NOTE This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 6.2 of ISO 13485:2016.

4.1.5 SOFTWARE ITEMS from third-party suppliers

The MANUFACTURER shall ensure that third-party suppliers perform applicable SECURITY LIFE CYCLE ACTIVITIES for each SOFTWARE ITEM if it meets both of the following criteria:

a) the SOFTWARE ITEM is mainly developed specifically for the MANUFACTURER and for a specific purpose; and

b) the SOFTWARE ITEM can have an impact on SECURITY.

The MANUFACTURER shall communicate requirements related to SECURITY for each SOFTWARE ITEM specifically developed by a third-party for the MANUFACTURER and for a specific purpose.

NOTE This requirement applies when the MANUFACTURER subcontracts a third-party to specifically develop a SOFTWARE ITEM which can have SECURITY implications. THREAT MODELLING is usually used to determine which SOFTWARE ITEM will have SECURITY implications.

4.1.6 Continuous improvement

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for continuously improving the SECURITY development LIFE CYCLE. This ACTIVITY (or ACTIVITIES) shall include the analysis of SECURITY defects in SOFTWARE ITEMS / HEALTH SOFTWARE / PRODUCTS that have been deployed to the field – due to insufficient or lacking ACTIVITIES.

NOTE 1 This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 8.5 of ISO 13485:2016.

NOTE 2 This ACTIVITY (or ACTIVITIES) ensures that the MANUFACTURER improves the rigor of their SECURITY ACTIVITIES over time. In case of PROCESS-dependent SECURITY defects, it is important for the MANUFACTURER to help compensate for this by continuously improving their SECURITY ACTIVITIES.

4.1.7 Disclosing SECURITY-related issues

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for informing regulatory authorities and PRODUCT users about VULNERABILITIES (that have been identified through ACTIVITIES as specified in 9.5) in supported PRODUCTS in a timely manner with content that includes but is not limited to the following information:

a) VULNERABILITY description, VULNERABILITY score as per CVSS or a similar system for ranking VULNERABILITIES, and affected PRODUCT version(s); and

b) description of the resolution.

NOTE 1 The description of the resolution can include references to installation of SECURITY updates – see Clause 12 of IEC 62443-4-1:2018[11].

NOTE 2 Timeliness is driven by authorities, applicable legislation, regulatory policy, PRODUCT SAFETY, and market forces. The strategy for handling third-party component VULNERABILITIES discovered by the PRODUCT developer takes into account the possibility of public disclosure by the third-party component supplier.

NOTE 3 This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 7.2.3 of ISO 13485:2016.

NOTE 4 See 4.1.9, 4.2 and 6.2.

4.1.8 Periodic review of SECURITY defect management

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for conducting periodic reviews of the software problem resolution PROCESS.

Periodic reviews of the ACTIVITIES shall, at a minimum, examine SECURITY-related issues managed through the PROCESS since the last periodic review to determine if the management PROCESS was complete, efficient, and led to the resolution of SECURITY-related issues.

Periodic reviews of the SECURITY-related issue management PROCESS shall be conducted at least annually or as part of monitoring, measurement and analysis of PROCESSES of 4.1.3 of ISO 13485:2016.

NOTE This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 5.6 of ISO 13485:2016.

4.1.9 ACCOMPANYING DOCUMENTATION review

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for identifying, characterizing and tracking to closure SECURITY-related errors and omissions in ACCOMPANYING DOCUMENTATION including the SECURITY guidelines.

NOTE This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 7.3 of ISO 13485:2016.

4.2 SECURITY RISK MANAGEMENT

The MANUFACTURER shall establish a PROCESS for managing RISKS associated with SECURITY. This PROCESS shall use THREAT MODELLING for identifying VULNERABILITIES, estimating and evaluating the associated THREATS, controlling these THREATS, and monitoring the effectiveness of the RISK CONTROL (SECURITY) measures, taking into account the INTENDED USE and the USE ENVIRONMENT of the HEALTH SOFTWARE.

NOTE 1 A medical device designed in a layered DEFENSE-IN-DEPTH approach does not rely on SECURITY controls in the operating environment. Nevertheless, as part of this layered DEFENSE-IN-DEPTH approach, there are expectations on the intended operating environment. Expectations on the operating environment can include protection and performance characteristics and can be inputs to THREAT MODELLING.

The MANUFACTURER shall establish the criteria for risk acceptability that shall be applied when determining the appropriate way to address each VULNERABILITY.

The SECURITY RISK MANAGEMENT should incorporate outcomes of the THREAT MODELLING ACTIVITY (or ACTIVITIES) and follow guidelines and industry best practice.

Detailed PROCESS steps are described in Clause 7.

NOTE 2 This PROCESS can be part of an existing general RISK MANAGEMENT PROCESS. SECURITY RISK MANAGEMENT can be conducted under the framework of ISO 14971 with an appropriate mapping of VULNERABILITY, THREAT and other SECURITY-related terms and addition of SECURITY-relevant ACTIVITIES. (See ISO TR 24971:2020 for possible mapping.)

The MANUFACTURER shall document any RESIDUAL RISK associated with a VULNERABILITY that remains in the system and shall also document respective compensating controls applied.

See Annex C on THREAT MODELLING.

4.3 SOFTWARE ITEM classification relating to risk transfer

The MANUFACTURER shall document which SOFTWARE ITEM is either

a) MAINTAINED SOFTWARE;

b) SUPPORTED SOFTWARE; or

c) REQUIRED SOFTWARE.

NOTE This ACTIVITY (or ACTIVITIES) can be implemented for example as a part of 7.4 of ISO 13485:2016.

5 Software development PROCESS

5.1 Software development planning

5.1.1 ACTIVITIES in the LIFE CYCLE PROCESS

The MANUFACTURER shall establish general LIFE CYCLE ACTIVITIES – from conception to decommissioning – that are consistent and integrated with a commonly accepted PRODUCT development PROCESS including but not limited to:

a) CONFIGURATION MANAGEMENT with change controls and change history;
b) PRODUCT description and requirements definition with requirements TRACEABILITY;
c) software or hardware design and implementation practices, such as modular design;
d) repeatable testing VERIFICATION and VALIDATION PROCESS;
e) review and approval of all development PROCESS records;
f) PRODUCT support; and
g) SECURITY updates and patching for HEALTH SOFTWARE.

NOTE PRODUCT support means providing of information, assistance and training to install and make HEALTH SOFTWARE operational in its intended environment and to distribute improved capabilities to users. See ISO/IEC/IEEE 24765:2017.

The MANUFACTURER shall document the justification for not implementing requirements of this document within a given HEALTH SOFTWARE project based on review and approval by personnel with the appropriate SECURITY expertise.

5.1.2 Development environment SECURITY

The MANUFACTURER shall establish risk-based procedural and technical controls for protecting the IT infrastructure used for development, production delivery and maintenance from unauthorized access, corruption and deletion. This includes protecting the HEALTH SOFTWARE during design, implementation, updates, testing and release.

5.1.3 Secure coding standards

The MANUFACTURER shall establish and maintain secure coding standards consistent with current best practices related to the design and implementation of secure software systems.

See Annex A.

5.2 HEALTH SOFTWARE requirements analysis

5.2.1 HEALTH SOFTWARE SECURITY requirements

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for ensuring that SECURITY requirements are documented for the HEALTH SOFTWARE including requirements for SECURITY CAPABILITIES related to installation, operation, maintenance and decommissioning.

NOTE 1 IEC TR 60601-4-5 gives guidance on the specification of SECURITY CAPABILITIES and their documentation in the ACCOMPANYING DOCUMENTATION and provides a method of determining requirements from the SECURITY CAPABILITY level.

NOTE 2 IEC TR 80001-2-2 specifies SECURITY-related needs, risks and controls as a guidance for disclosure and communication between the MANUFACTURER and the HEALTHCARE DELIVERY ORGANIZATION.

NOTE 3 The PRODUCT requirements PROCESS interfaces with HEALTH SOFTWARE requirements. Some technical controls can be implemented at PRODUCT level (for example by hardware). See E.2.1.

5.2.2 SECURITY requirements review

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for ensuring that SECURITY requirements

a) implement PRODUCT requirements including those relating to RISK CONTROL;
b) do not contradict one another;
c) are expressed in terms that avoid ambiguity; and
d) are stated in terms that permit establishment of test criteria and performance of tests.

The MANUFACTURER shall document the level of independence of the reviewers. Each of the following representative disciplines shall participate in this ACTIVITY (or ACTIVITIES):

a) architects/developers (those who will implement the requirements);
b) testers (those who will validate that the requirements have been met);
c) cross-functional experts (can include those with clinical expertise); and
d) SECURITY advisor(s).

NOTE 1 A single person can be responsible for multiple disciplines. It is not advisable to have a single person representing all disciplines.

NOTE 2 The list of disciplines is documented at least once per project.

NOTE 3 A quality management system like that of ISO 13485 implies consideration of independence of reviewers.

5.2.3 SECURITY risks for REQUIRED SOFTWARE

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) that identifies and manages the SECURITY risks of all REQUIRED SOFTWARE.

NOTE 1 This ACTIVITY (or ACTIVITIES) ensures that HEALTH SOFTWARE requirements are aware of the SECURITY needs of REQUIRED SOFTWARE.

NOTE 2 This ACTIVITY(or ACTIVITIES) can be part of supply chain SECURITY ACTIVITIES.

5.3 Software architectural design

5.3.1 DEFENSE-IN-DEPTH ARCHITECTURE/design

The MANUFACTURER shall establish an ACTIVITY(or ACTIVITIES) to specify a secure ARCHITECTURE.

At each stage of development, the MANUFACTURER should consider DEFENSE-IN-DEPTH and assign technical requirements to each layer of defense.

When identifying technical SECURITY RISK CONTROLS, the MANUFACTURER shall take into account requirements regarding SAFETY or performance of HEALTH SOFTWARE.

NOTE DEFENSE-IN-DEPTH can include SECURITY requirements in the ACCOMPANYING DOCUMENTATION to be implemented by the HDO.

5.3.2 Secure design best practices

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to identify, enforce and maintain secure design practices. The MANUFACTURER shall document secure design best practices, which should include but are not limited to:

a) documenting all TRUST BOUNDARIES as part of the design;
b) least privilege (granting only the privileges to users/software necessary to perform intended operations);
c) using proven secure SOFTWARE ITEMS/designs where possible;

d) economy of mechanism (striving for simple designs);

e) using secure design patterns;

f) ATTACK SURFACE reduction;

g) removing backdoors, debug access and debug information used during development or documenting their presence and the need to protect them from unauthorized access; and

h) protecting any remaining debug information from unauthorized access.

The MANUFACTURER shall define a SECURITY ARCHITECTURE as part of DEFENSE-IN-DEPTH, including the practices listed above as appropriate.

NOTE See Annex B.

5.3.3 SECURITY architectural design review

The MANUFACTURER shall implement an architectural review of the HEALTH SOFTWARE with respect to behavior under adverse conditions:

a) effective segregation of SOFTWARE ITEMS;

b) the secure design best practices (see 5.3.2); and

c) potential SECURITY flaws introduced by the ARCHITECTURE.

The MANUFACTURER shall document and implement the architectural design review.

NOTE Segregation uses technical controls in design and implementation in order to ensure that SOFTWARE ITEMS cannot be influenced by other SOFTWARE ITEMS of the HEALTH SOFTWARE in an unintended way.

5.4 Software design

5.4.1 Software design best practices

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to develop and document a secure HEALTH SOFTWARE design and maintain the use of best practices for the secure design, taking into account:

a) software technology at application level (for examples algorithms, methods);

b) the programming technology used, (for example programming language);

c) the secure design best practices in 5.3.2.

5.4.2 Secure design

The HEALTH SOFTWARE design shall include measures to address the THREATS identified in the THREAT MODEL.

NOTE The SECURITY CONTEXT for HEALTH SOFTWARE is derived from the INTENDED ENVIRONMENT OF USE at PRODUCT-level, considering also the configuration and integration of HEALTH SOFTWARE.

5.4.3 Secure HEALTH SOFTWARE interfaces

The HEALTH SOFTWARE design shall identify and characterize each interface of the HEALTH SOFTWARE including physical and logical interfaces. As appropriate, the MANUFACTURER identifies as part of the design:

a) whether the interface is externally accessible (by other PRODUCTS) or internally accessible – between SOFTWARE ITEMS of the HEALTH SOFTWARE- or both;

b) SECURITY implications of the HEALTH SOFTWARE SECURITY CONTEXT on the external interface;

c) potential users of the interface and the ASSETS that can be accessed through the interfaces (directly or indirectly);

d) whether the static design includes access to interfaces across TRUST BOUNDARIES;

e) SECURITY considerations, assumptions and/or constraints associated with the use of the interface within the HEALTH SOFTWARE SECURITY CONTEXT; including applicable THREATS;

f) the SECURITY roles, privileges/rights and access control permissions needed to use the interface and to access the ASSETS defined in c);

g) the SECURITY CAPABILITIES and/or compensating mechanisms used to safeguard the interface and the ASSETS identified in c) including run-time VALIDATION of inputs as well as handling outputs and errors;

h) the use of third-party SOFTWARE ITEMS to implement the interface and their SECURITY CAPABILITIES;

i) documentation that describes how to use the interface if it is externally accessible; and

j) description of how the design mitigates the THREATS identified in the THREAT MODEL.

NOTE The SECURITY CONTEXT for HEALTH SOFTWARE is derived from the INTENDED ENVIRONMENT OF USE at PRODUCT-level, considering also the configuration and integration of HEALTH SOFTWARE.

5.4.4 Detailed design VERIFICATION for SECURITY

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for conducting design reviews to identify, characterize and track to closure WEAKNESSES associated with each significant revision of the secure design including but not limited to:

a) SECURITY requirements that were not adequately addressed by the design;

b) THREATS and their ability to exploit VULNERABILITIES in PRODUCT interfaces, TRUST BOUNDARIES and ASSETS;

c) identification, documentation and characterization of detailed design best-practices that were not followed (5.3.2 and 5.4.1).

NOTE The design reviews also take into account each software service that is used by HEALTH SOFTWARE to achieve its intended functionality, for example: cloud, software-/ infrastructure-/ platform-as-a-service.

5.5 Software unit implementation and VERIFICATION

5.5.1 Secure coding standards

The MANUFACTURER shall establish an implementation ACTIVITY (or ACTIVITIES) following secure coding standards.

See clause A.4.

5.5.2 SECURITY implementation review

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to ensure that implementation reviews are performed for identifying, characterizing and feeding into the problem resolution PROCESS all SECURITY-related issues associated with the implementation of the secure design including:

a) identification of SECURITY requirements (see 5.2) that were not adequately addressed by the implementation;

   NOTE Requirements allocation, including SECURITY requirements, is part of typical design PROCESSES.

b) identify secure coding standards used and document any parts of the secure coding standards that were not followed (for example, use of banned functions or failure to apply the principle of least privilege);

c) Static Code Analysis (SCA) for source code to determine secure coding errors using the secure coding standard for the supported programming language, as established in 5.1.3. SCA is often supported by tools, but it can be done through code inspections and code-walkthroughs.

d) review of the implementation and its TRACEABILITY to the SECURITY CAPABILITIES defined to support the SECURITY design (see 5.3 and 5.4); and

e) examination of THREATS and their ability to exploit implementation interfaces, TRUST BOUNDARIES and ASSETS (see 5.3 and 5.4).

5.6 Software integration testing

The MANUFACTURER can perform some of the software system testing as a part of software integration testing (see 5.7).

As a part of HEALTH SOFTWARE integration testing, the MANUFACTURER should consider SECURITY policy differences across TRUST BOUNDARIES.

5.7 Software system testing

5.7.1 SECURITY requirements testing

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for verifying that the HEALTH SOFTWARE SECURITY functions meet the SECURITY requirements and that the HEALTH SOFTWARE handles error scenarios and invalid input. Based on the INTENDED ENVIRONMENT OF USE, types of testing shall include:

a) functional testing of SECURITY requirements;
b) performance and scalability testing;
c) boundary/edge condition, stress and malformed or unexpected input tests with potential SECURITY consequences; and
d) testing each software service that is used by HEALTH SOFTWARE to achieve its intended functionality, in the context of responsibility agreements among service providers, MANUFACTURERS and operators, for example: cloud services, software-as-a-service, infrastructure-as-a-service, platform-as-a-service.

NOTE See B.5.7.1.

5.7.2 THREAT mitigation testing

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for testing the effectiveness of the mitigation for the THREATS identified and assessed in the THREAT MODEL. ACTIVITIES shall include:

a) creating and executing adequate testing for each mitigation implemented to address a specific THREAT, in order to ensure that the mitigation works as designed;
b) creating and executing plans for attempting to thwart each mitigation; and
c) ensuring that the mitigation does not introduce other VULNERABILITIES to the design.

5.7.3 VULNERABILITY testing

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for performing tests that focus on identifying and characterizing potential SECURITY VULNERABILITIES in the HEALTH SOFTWARE. Known VULNERABILITY testing shall be based upon, at a minimum, recent contents of an established, industry-recognized, public source for known VULNERABILITIES. As appropriate, testing shall include:

a) abuse case for malformed or unexpected input testing focused on uncovering SECURITY issues. This shall include manual or automated abuse case testing and specialized types of abuse case testing on all external interfaces and protocols. Examples include fuzz testing and network traffic load testing and capacity testing;
b) ATTACK SURFACE testing to determine all avenues of ingress and egress to and from the system, common VULNERABILITIES including but not limited to weak access-control-lists (ACLs), exposed ports and services running with elevated privileges;

c) "closed box" known VULNERABILITY scanning focused on detecting known VULNERABILITIES in (if applicable) hardware, host, interfaces or SOFTWARE ITEMS;

NOTE 1 For example, this could be a network based known VULNERABILITY scan.

d) SOFTWARE COMPOSITION ANALYSIS on all binary executable files including embedded firmware, to be used with HEALTH SOFTWARE and delivered by a third-party supplier. This analysis can be used to detect:

   1) known VULNERABILITIES in the SOFTWARE ITEMS;
   2) linking to vulnerable libraries;
   3) SECURITY rule violations;
   4) compiler settings that can lead to VULNERABILITIES; and
   5) comparison of the software encountered to the software bill of materials.

   NOTE 2 Tools can support SOFTWARE COMPOSITION ANALYSIS by generating a list of software packages included.

e) dynamic SECURITY testing – like e.g. fuzz testing, that detects flaws not visible under static code analysis, including but not limited to denial of service conditions due to failing to release runtime handles, memory leaks and accesses made to shared memory without authentication. This testing shall be applied if such tools are available.

NOTE 3 Exhaustive runtime testing cannot be done effectively without tools.

5.7.4 Penetration testing

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to identify and characterize WEAKNESSES via tests that focus on discovering and exploiting SECURITY VULNERABILITIES in the HEALTH SOFTWARE.

See B.5.7.4.

5.7.5 Managing conflicts of interest between testers and developers

The MANUFACTURER shall document means of ensuring objectivity of the test effort for these tests:

a) ATTACK SURFACE analysis,
b) SECURITY requirements testing,
c) THREAT mitigation testing,
d) VULNERABILITY testing,
e) known VULNERABILITY scanning and
f) penetration testing.

NOTE Objectivity intends to support reproducibility of outcomes based on facts.

See A.1 and B.5.7.5.

5.8 Software release

5.8.1 Resolve findings prior to release

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to ensure that all findings from system testing have been handled by the problem resolution PROCESS (Clause 9).

5.8.2 Release documentation

As a part of the software release ACTIVITY (or ACTIVITIES), the MANUFACTURER shall establish requirements for ACCOMPANYING DOCUMENTATION:

a) secure operation guidelines;
b) PROCESS rigor and conformance documentation including the scoping (Clause 4), tailoring (Clause 5) and information on coverage of documentation (Annex E);

   NOTE These documents help meet any regulatory or contractual obligations.

c) account management guidelines (if applicable); and
d) appropriate information about relevant RESIDUAL RISKS to SECURITY remaining in the HEALTH SOFTWARE.

See Annex E for an informative specification of documentation contents.

5.8.3 File INTEGRITY

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to provide an INTEGRITY VERIFICATION mechanism for all scripts, executables and other SECURITY-relevant files used with a HEALTH SOFTWARE.

This ACTIVITY (or ACTIVITIES) is required to ensure that PRODUCT users can verify that executables, scripts, and other important files received from the MANUFACTURER have not been altered. Common methods of meeting this requirement include cryptographic hashes and digital signatures (which also provide proof of origin).

5.8.4 Controls for private keys

The MANUFACTURER shall have procedural and technical controls in place to protect private keys used for code signing from unauthorized access or modification.

NOTE This refers to the software supply chain and the focus is on code signing to support secure distribution and delivery of HEALTH SOFTWARE.

5.8.5 Assessing and addressing SECURITY-related issues

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for verifying that a HEALTH SOFTWARE or an update is not released until its SECURITY-related issues have been addressed and tracked to closure (see 9.5). This includes issues associated with:

a) requirements (see 5.2);
b) SECURITY by design (see 5.3 and 5.4);
c) implementation (see 5.5);
d) VERIFICATION / VALIDATION (see 5.5, 5.6 and 5.7); and
e) SECURITY defect management (see 9.4).

5.8.6 ACTIVITY completion

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for verifying that, prior to HEALTH SOFTWARE release, all applicable SECURITY-related PROCESSES required by this document have been completed with records documenting the completion of each ACTIVITY (or ACTIVITIES) or PROCESS.

5.8.7 SECURE decommissioning guidelines for HEALTH SOFTWARE

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to create PRODUCT user documentation that includes guidelines for removing the HEALTH SOFTWARE from use.

6 SOFTWARE MAINTENANCE PROCESS

6.1 Establish SOFTWARE MAINTENANCE plan

6.1.1 Timely delivery of SECURITY updates

The MANUFACTURER shall establish – as a part of the update ACTIVITIES – a policy that specifies the timeframes for delivering and qualifying SECURITY updates to PRODUCT users. At a minimum, this policy shall consider the following factors:

a) the potential impact (technical, for SAFETY, effectiveness, SECURITY) of the VULNERABILITY;
b) public knowledge of the VULNERABILITY;
c) whether published EXPLOITS exist for the VULNERABILITY;
d) the volume of deployed PRODUCTS that are affected; and
e) the AVAILABILITY of an effective external control when no HEALTH SOFTWARE update is being provided.

NOTE 1 Some regulatory authorities can have specific timeframe requirements.

NOTE 2 The MANUFACTURER can categorize SECURITY updates (for example by potential impact) and specify appropriate timeframes. See IEC TR 60601-4-5.

NOTE 3 During an acceptable time-interval in which the MANUFACTURER develops a technical control, any documented mitigations and constraints on the INTENDED USE can be based on RISK MANAGEMENT. It is advisable to develop and deploy a technical mitigation in HEALTH SOFTWARE.

NOTE 4 IEC TR 60601-4-5 specifies a minimum performance ("Essential Function" term as used in IEC 62443 series) to be available with medical devices in case of relevant CYBERSECURITY ATTACKS on the HEALTHCARE DELIVERY ORGANIZATION'S (HDO) IT network. Such minimum performance ensures basic functionality until a verified SECURITY update is available in situations in which all medical devices of the same type in the HDO can be affected by a given CYBERSECURITY ATTACK simultaneously. Therefore, for medical devices, the software LIFE CYCLE ACTIVITIES ensure that:

a) "essential functions" remain secure for the interval until a SECURITY update is installed, and
b) SECURITY updates always re-establish the SECURITY CAPABILITY as specified in the ACCOMPANYING DOCUMENTATION.

Additional guidance is provided by IEC TR 60601-4-5.

6.2 Problem and modification analysis

6.2.1 Monitoring public incident reports

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to actively collect and review relevant sources of information about VULNERABILITIES regarding SUPPORTED SOFTWARE.

NOTE This includes HEALTH SOFTWARE.

6.2.2 SECURITY update VERIFICATION

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for verifying that SECURITY updates created by the MANUFACTURER address the intended SECURITY VULNERABILITIES.

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for verifying that SECURITY updates do not introduce unintended effects to functional or quality attributes of HEALTH SOFTWARE. Such SECURITY updates include but are not limited to updates created by:

a) the HEALTH SOFTWARE MANUFACTURER,
b) suppliers of SOFTWARE ITEMS used in the HEALTH SOFTWARE, and
c) suppliers of SOFTWARE ITEMS or platforms on which the HEALTH SOFTWARE depends.

The MANUFACTURER can define that for certain SOFTWARE ITEMS or platforms, there is a shared responsibility for such VERIFICATION.

NOTE Also see Clause 9.

6.3 Modification implementation

6.3.1 SUPPORTED SOFTWARE SECURITY update documentation

The MANUFACTURER shall establish a policy to inform PRODUCT users about updates for SUPPORTED SOFTWARE. This information shall include:

a) stating whether the HEALTH SOFTWARE is compatible with the SUPPORTED SOFTWARE SECURITY update; and

b) for SECURITY updates that are unapproved by the HEALTH SOFTWARE MANUFACTURER, the mitigations that can be used instead of applying the update.

6.3.2 MAINTAINED SOFTWARE SECURITY update delivery

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to ensure that SECURITY updates are made available for MAINTAINED SOFTWARE to PRODUCT users.

See E.2.5.

6.3.3 MAINTAINED SOFTWARE SECURITY update INTEGRITY

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to ensure that each applicable update for MAINTAINED SOFTWARE is made available to PRODUCT users in a manner that facilitates INTEGRITY VERIFICATION of the SECURITY update.

This ACTIVITY (or ACTIVITIES) is required to ensure that HEALTH SOFTWARE users can obtain applicable SECURITY patches for the MAINTAINED SOFTWARE to reduce the possibility that the SECURITY patches are fraudulent. Having this ACTIVITY (or ACTIVITIES) means that the MANUFACTURER provides a mechanism or technique that allows HEALTH SOFTWARE users to verify the authenticity of patches. Concurrent release of patches for all MAINTAINED SOFTWARE can reduce the time window between awareness of the VULNERABILITY and the AVAILABILITY of patches.

7 SECURITY RISK MANAGEMENT PROCESS

7.1 RISK MANAGEMENT context

7.1.1 General

The MANUFACTURER shall establish and maintain a PROCESS for managing SECURITY risks related to HEALTH SOFTWARE as a part of its PRODUCT RISK MANAGEMENT approach. This PROCESS should consist of the following PROCESS steps described in 7.1.2, 7.2, 7.3, 7.4 and 7.5.

See Annex C.

7.1.2 PRODUCT SECURITY CONTEXT

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to ensure that the intended PRODUCT SECURITY CONTEXT is documented. This ACTIVITY (or ACTIVITIES) is required to ensure that the minimum requirements of the environment and the assumptions about that environment are documented in order to achieve the SECURITY level for which the PRODUCT was designed.

The purpose of defining this information is so that both the developers of the HEALTH SOFTWARE and the PRODUCT users have the same understanding about how the PRODUCT is intended to be used. This will help the developers make appropriate design decisions and the users to use the PRODUCT as it was intended.

The SECURITY CONTEXT could include:

a) location in the network;
b) physical SECURITY or CYBERSECURITY provided by the environment where the PRODUCT will be deployed;
c) isolation (from a network perspective);
d) if known, potential impact to SAFETY caused by degradation of SECURITY;
e) SECURITY controls implemented in dedicated hardware with which the HEALTH SOFTWARE is intended to be used.

For example, it is important to document whether physical SECURITY is required. If no physical SECURITY is expected to be present, then that can add a number of related requirements such as not allowing push-button configuration on the PRODUCT. Another example is if the PRODUCT cannot feasibly (i.e. without reducing SAFETY or performance) implement a firewall of its own, it can be expected to be protected by a user-supplied firewall that connects it to the health-IT-network.

Documenting these external SECURITY features for the PRODUCT (its SECURITY CONTEXT) allows developers to design a DEFENSE-IN-DEPTH strategy that complements this SECURITY CONTEXT and testers to validate and verify the SECURITY of a PRODUCT in an environment similar to how it is intended to be deployed.

Having this PROCESS means that the deployment environment in which the PRODUCT is intended to be used is correctly represented in all PROCESSES involved in the development and testing of this PRODUCT and are documented.

7.2 Identification of VULNERABILITIES, THREATS and associated adverse impacts

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) which identifies and documents any vulnerabilities, THREATS and associated adverse impacts affecting CONFIDENTIALITY, INTEGRITY, AVAILABILITY of ASSETS in HEALTH SOFTWARE. This ACTIVITY (or ACTIVITIES) shall consider the INTENDED USE and the INTENDED ENVIRONMENT OF USE with respect to the SECURITY CONTEXT.

This ACTIVITY (or ACTIVITIES) shall be employed to ensure that all PRODUCTS shall have a THREAT MODEL specific to the current development scope of the PRODUCT with the following characteristics (where applicable):

a) correct flow of categorized information throughout the system;
b) TRUST BOUNDARIES;
c) PROCESSES;
d) data stores;
e) interacting external entities;
f) internal and external communication protocols implemented in the PRODUCT;
g) externally accessible physical ports including debug ports;
h) circuit board connections such as Joint Test Action Group (JTAG) connections or debug headers which might be used to ATTACK the hardware;
i) potential ATTACK vectors including ATTACK on the (intended) hardware;
j) potential THREATS;
k) SECURITY-related issues identified; and

l) external dependencies in the form of drivers or third-party applications (code that is not developed by the supplier) that are linked into the application.

The THREAT MODEL shall be reviewed and verified by the development team to ensure that it is correct and understood.

The THREAT MODEL shall be reviewed periodically (at least once a year) for released PRODUCTS and updated if required in response to the emergence of new THREATS to the PRODUCT even if the design does not change.

Any issues identified in the THREAT MODEL shall be addressed as defined in 9.4 and 9.5.

7.3 Estimation and evaluation of SECURITY risk

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to:

a) estimate the risk of the VULNERABILITIES identified above. Risk estimation is done considering the adverse impact of that VULNERABILITY to CYBERSECURITY. This estimation can be supported by using VULNERABILITY scoring, such as the Common VULNERABILITY Scoring System (CVSS) or MITRE[30] scoring rubric for medical devices. The scoring system can also be based on a likelihood/severity scheme used by the MANUFACTURER for other risks (see e.g. ISO/IEC Guide 51 or ISO 14971);
b) evaluate the estimated risks and – based on scoring – determine if the risk is acceptable or not; and
c) inform the PRODUCT RISK MANAGEMENT PROCESS about any updates to the THREAT MODEL.

7.4 Controlling SECURITY risks

The MANUFACTURER shall determine whether SECURITY RISK CONTROL measures are appropriate for reducing the SECURITY risks to an acceptable level based on SECURITY risk acceptance policies. If RISK CONTROLS are deemed appropriate, the MANUFACTURER shall:

- select appropriate mitigations;
- determine whether these mitigations result in new risks or increase other risks;
- implement selected mitigations; and
- verify the effectiveness of the implemented measures.

The MANUFACTURER shall document the results of these ACTIVITIES.

Handling of RESIDUAL RISKS to SECURITY shall be done in cooperation with the PRODUCT RISK MANAGEMENT.

NOTE The assessment of SECURITY RISKS is influenced by the SECURITY CONTEXT. The SECURITY RISK acceptability is based on the respective score and the acceptability threshold for SECURITY RISKS. Also see 4.2.

7.5 Monitoring the effectiveness of RISK CONTROLS

The MANUFACTURER shall monitor the effectiveness of RISK CONTROLS by information collection and review during the post-release phase.

This ACTIVITY (or ACTIVITIES) shall also inform other ACTIVITIES and PROCESSES of the issue or related issue(s), including PROCESSES for other PRODUCTS / revisions; and inform third parties (e.g. suppliers) if problems have been found in third-party source code to be used with the HEALTH SOFTWARE.

Any issues identified in the THREAT MODEL of released HEALTH SOFTWARE will be addressed as defined in 9.4 and 9.5.

8 Software CONFIGURATION MANAGEMENT PROCESS

The MANUFACTURER shall establish a general PRODUCT development/maintenance/support PROCESS that includes CONFIGURATION MANAGEMENT with change controls and change history.

For SECURITY obligations with HEALTH SOFTWARE already released or in the market, CONFIGURATION MANAGEMENT shall provide the capability to reproduce a list of included external components that are or could become susceptible to VULNERABILITIES.

9 Software problem resolution PROCESS

9.1 Overview

The ACTIVITIES specified by this clause are used for handling SECURITY-related issues of HEALTH SOFTWARE.

9.2 Receiving notifications about VULNERABILITIES

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) that enables the reporting of information regarding VULNERABILITIES to the MANUFACTURER – independent of whether they come from an internal entity, an external entity or via a complaint-handling system.

This reception ACTIVITY (or ACTIVITIES) shall receive and track to closure reports on SECURITY-related issues in the HEALTH SOFTWARE from the following sources including at a minimum:

a) SECURITY VERIFICATION and VALIDATION testers;
b) suppliers of third-party components used in the PRODUCT;
c) PRODUCT developers and testers;
d) PRODUCT users including integrators, operators, administrators, and maintenance personnel;
e) data obtained from audit event log information;
f) SECURITY researchers (SECURITY VULNERABILITY reporters), also see ISO/IEC 29147; and
g) data or notifications about widespread VULNERABILITIES that can affect the HEALTH SOFTWARE – See 6.2.

NOTE Typically, such information comes from publications, reports, independent SECURITY research, internal investigations, CERTs and Information Sharing and Analysis Organizations (ISAOs).

9.3 Reviewing VULNERABILITIES

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) that enables the investigation of VULNERABILITIES in a timely manner to determine their:

a) applicability to the PRODUCT;
b) verifiability; and
c) related THREATS.

NOTE 1 Timeliness is driven by authorities, applicable legislation, regulatory policy and market forces.

NOTE 2 This PROCESS can be implemented for example as a part of the PROCESSES per 8.2.1, 8.2.2 and 8.2.3 of ISO 13485:2016.

9.4 Analysing VULNERABILITIES

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) for analysing VULNERABILITIES in the PRODUCT to include:

a) assessing their impact with respect to:
   1) the technical SECURITY CONTEXT in which they were discovered; (see Clause 6 of IEC 62443-4-1:2018[11]);
   2) the PRODUCT'S INTENDED ENVIRONMENT OF USE, and
   3) the PRODUCT'S DEFENSE-IN-DEPTH strategy;
b) impact as defined by a VULNERABILITY scoring system (for example CVSS);
c) identifying all other PRODUCTS / PRODUCT versions containing the SECURITY-related issue (if any);
d) identifying the root cause of the issue;
e) identifying related SECURITY issues (that is, in the same PRODUCT); and
f) impact on PRODUCT SAFETY and effectiveness.

NOTE 1 For root cause analysis, a methodical approach such as described in IEC 62740 can be employed.

NOTE 2 A root cause is the first event in a sequence of causal factors which is deviating from the intended sequence.

NOTE 3 Not all root causes can be fixed by technical measures in HEALTH SOFTWARE.

NOTE 4 This PROCESS can be implemented for example as a part of 8.5.2 of ISO 13485:2016 and 8.5.3 of ISO 13485:2016.

9.5 Addressing SECURITY-related issues

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to address SECURITY-related issues and determine whether to disclose them (under 4.1.7) based on the results of the impact assessment and the acceptable level of RESIDUAL RISK.

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to determine whether and how identified SECURITY risks will be handled – via the problem resolution PROCESS or through updated specifications regarding the INTENDED ENVIRONMENT OF USE.

The MANUFACTURER shall establish an ACTIVITY (or ACTIVITIES) to review any changes to the design or implementation for impact on SAFETY, SECURITY and effectiveness.

The MANUFACTURER shall inform other PROCESSES of the issue or related issue(s), including PROCESSES for other PRODUCTS / PRODUCT revisions. This can be done by submitting problem reports or similar into other PROCESSES.

The MANUFACTURER shall inform third parties if problems have been found in third-party source code to be used with SUPPORTED HEALTH SOFTWARE. In case of open-source software, the publishing platform can be used to inform about or fix the issue found.

This ACTIVITY shall include a periodic review of open SECURITY-related issues to ensure that issues are being addressed appropriately. This periodic review shall at a minimum occur during each PRODUCT release; see 4.1.6 and 4.1.8.

NOTE 1 This periodic review can be implemented for example as a part of 8.2.6 of ISO 13485:2016.

NOTE 2 As an example, an intended function including the transmission of personally identifiable information through an external network can raise the need for data encryption.

NOTE 3

- For some THREATS it can be feasible to not mitigate them through technical measures in HEALTH SOFTWARE, because they can be linked to the INTENDED USE or essential functions.
  - Example: Console access to emergency / acute care devices would be hindered by overly complex authentication procedures and might delay the delivery of urgent care.
  - Example: Strong cryptography algorithms for encrypting data used for near-field transmission in principle use considerable computing power and can drain the battery when implemented in smaller, mobile devices.
- Some THREATS can be better addressed by mitigations in the INTENDED ENVIRONMENT OF USE which are expressed via ACCOMPANYING DOCUMENTATION (see Annex E).
- There are VULNERABILITIES that cannot be exploited because of measures in the design of the HEALTH SOFTWARE.

NOTE 4 Because of the complexity in determining the probability related to THREATS, the concept of likelihood is more appropriate and commonly used for IT- SECURITY. Likelihood of identified THREATS is typically expressed through structured scoring systems like Common VULNERABILITY Scoring Systems (CVSS), which can also take into account the attacker’s gain in relation to the applicable effort.

NOTE 5 RISK MANAGEMENT for medical device SAFETY – as in ISO 14971 – can be supported by a THREAT MODELLING method to cover SECURITY THREATS.

NOTE 6 IEC TR 63069 explains the relationship between the SAFETY / SECURITY PROCESSES.

Annex A
(informative)

Rationale

A.1 Relationship to IEC 62443

IEC 62443 is a series of Industrial Automation and Controls Systems SECURITY specifications. This series is the successor of ISA-99 and a well-recognized set of SECURITY standards for operational technology. Parts of the IEC 62443 series are recognized by the FDA, furthermore the EU "MDCG 2019-16 Guidance on Cybersecurity for medical devices" and the German BSI Guidance CS 132 refer to the IEC 62443 specifications.

Industrial Automation and Control Systems (IACS) recognize SECURITY as well as SAFETY and effectiveness. These are key properties that can also be applied in the field of HEALTH SOFTWARE.

Risk is a term used to describe the potential of damage towards certain protection goals. As an example, for medical devices these goals are the safe and performant operation of devices. In standards related to medical devices, risk is defined following 3.9 of ISO/IEC Guide 51:2014 or 3.10 of ISO/IEC Guide 63 as "combination of the probability of occurrence of harm and the severity of that harm" (with harm being defined as damage to *external* ASSETS like health, environment or property), while standards in the IEC 62443 series use the term risk to describe the potential of reduction of CONFIDENTIALITY, INTEGRITY, AVAILABILITY of rather system-*internal* ASSETS. However, the perspective of IEC 62443 includes risks that have the potential to cause *external* harm in the sense of ISO/IEC Guide 51 or ISO/IEC Guide 63. As this document dominantly uses this wider concept of (SECURITY) risk, we will not give a definition of the term as to avoid confusion with the more specific use of that term.

Due to the wide spectrum of technologies and applications for HEALTH SOFTWARE, it is difficult to prescribe a specific set of SECURITY CONTROLS. Practice in industry has shown that SECURITY measures in the LIFE CYCLE PROCESSES lead to secure PRODUCTS. This document therefore addresses LIFE CYCLE PROCESSES in responsibility of the MANUFACTURER and takes the requirements of IEC 62443-4-1[11], in consideration that the requirements:

- are relevant for HEALTH SOFTWARE;
- specify PROCESS-related requirements;
- address the MANUFACTURER;
- do not specify PRODUCT capabilities;
- do not specify documentation content for ACCOMPANYING DOCUMENTATION – which will be specified by IEC TR 60601-4-5.

Satisfying the requirements of this document will support conformance towards IEC 62443-4-1[11]. However, this document contains some adaptions and clarifications to the healthcare sector.

MANUFACTURERS striving at full conformance to IEC 62443-4-1[11] consider the following additional provisions regarding the independence from the developers "who designed and implemented the PRODUCT according to Table A.1.

Table A.1 – Required level of independence of testers from developers

Test type	Reference	Level of independence
SECURITY requirements testing	SVV-1 SECURITY requirements testing	Independent department
THREAT mitigation testing	SVV-2 THREAT mitigation testing	Independent department
VULNERABILITY testing	SVV-3 VULNERABILITY testing	Independent person
Static code analysis	SI-1 SECURITY implementation review	None
ATTACK SURFACE analysis	SVV-3 VULNERABILITY testing	Independent person
Known VULNERABILITY scanning	SVV-3 VULNERABILITY testing	Independent person
SOFTWARE COMPOSITION ANALYSIS	SVV-3 VULNERABILITY testing	None
Penetration testing	SVV-4 Penetration testing	Independent department or organization
SOURCE: IEC 62443-4-1:2018[11], SVV-5, 9.6.1.		

The MANUFACTURER can implement a (semi-)independent internal testing team and/or the use of a third-party SECURITY test organization. Individuals who are independent from the developers who designed and implemented the SECURITY features should do the SECURITY testing. The levels of independence can be “as follows:

- None – no independence required. Developer can perform the testing.
- Independent person – the person who performs the testing cannot be one of the developers of the PRODUCT.
- Independent department – the person who performs the testing cannot report to the same first line manager as any developers of the PRODUCT. Alternatively, they could be a member of a quality assurance (QA) department.
- Independent organization – the person who performs the testing cannot be part of the same organization as any developers of the PRODUCT. An organization can be a separate legal entity, a division of a company or a department of a company that reports to a different executive such as a vice president or similar level.”

For the use of this document, "Abuse case" in IEC 62443-4-1:2018[11], SVV-5, 9.6.1 was modified to "VULNERABILITY".

A.2 Relationship to IEC 62304

In order to extend existing LIFE CYCLE PROCESSES for HEALTH SOFTWARE, these requirements have been arranged in a structure reflecting that of IEC 62304[8].

Implementation of IEC 62304[8] is not required for implementing the PROCESSES specified in this document. However, if a MANUFACTURER identifies in their PROCESSES those ACTIVITIES specified in IEC 62304[8], it is easier to determine the related ACTIVITY (or ACTIVITIES) for CYBERSECURITY specified in this document.

IEC 62304[8] specifies ACTIVITIES, based on the software SAFETY classification. The required ACTIVITIES are indicated in the normative text of IEC 62304[8] as "[Class A, B, C]", "[Class B, C]" or "[Class C]", indicating that they are required selectively depending on the classification of the software to which they apply. The requirements in Clause 4 through Clause 9 of this document have a special focus on CYBERSECURITY and therefore do not follow the concept of SAFETY classes. For conformance to this document the selection of ACTIVITIES is independent of SAFETY classes.

A.3 Risk transfer

A.3.1 Overview

There are shared responsibilities for using HEALTH SOFTWARE in a secure way. As a part of deploying HEALTH SOFTWARE to the customer, some of the risk of secure operation is transferred, while some of the risk remains with the MANUFACTURER.

IEC 62443-4-1 does not clearly define terms or concepts for risk transfer, as it just notes that for "external provided components", "dependent components", and "custom development components" different requirements apply. Therefore, this document introduces categories for SOFTWARE ITEMS that declare different levels of transfer of responsibility and risk from the MANUFACTURER to the customer.

The MANUFACTURER will identify the following categories of risk transfer for each software that is required by HEALTH SOFTWARE to achieve its INTENDED PURPOSE.

A.3.2 MAINTAINED SOFTWARE

The MANUFACTURER will assume the risk related to the SECURITY of MAINTAINED SOFTWARE (6.3.2, 6.3.3). As a result, the MANUFACTURER will provide SECURITY updates for all software in this category.

Examples for MAINTAINED SOFTWARE include

- software from third party, specifically developed for use with HEALTH SOFTWARE;
- embedded off-the-shelf software; and
- HEALTH SOFTWARE including those developed prior to publication of this document.

A.3.3 SUPPORTED SOFTWARE

The MANUFACTURER will notify the customer about known risks related to the SECURITY of that software (6.3.1).

Examples for SUPPORTED SOFTWARE include

- generally available off-the-shelf software;
- software from third party, also intended for other uses than with HEALTH SOFTWARE; and
- MAINTAINED SOFTWARE.

A.3.4 REQUIRED SOFTWARE

The MANUFACTURER will assume known risks related to the SECURITY (5.2.1, 5.2.3) known before release of the software.

That means that all SECURITY requirements for each SOFTWARE ITEMS required by HEALTH SOFTWARE to achieve its INTENDED PURPOSE will be considered as part of the requirements specification of that HEALTH SOFTWARE.

Examples for REQUIRED SOFTWARE include

- software for which no updates can be provided;
- obsolete third-party software; and
- SUPPORTED SOFTWARE.

A.4 Secure coding best practices

The secure coding best practices for HEALTH SOFTWARE should include at a minimum:

a) avoidance of potentially exploitable implementation constructs – implementation design patterns that are known to have SECURITY WEAKNESSES,

b) avoidance of banned functions and coding constructs/design patterns – software functions and design patterns that should not be used because they have known SECURITY WEAKNESSES,

   NOTE 1 Best coding practices avoid coding based on unspecified or undefined behaviour of the programming environment.

   NOTE 2 For common libraries and programming languages there are public lists of banned functions. Per secure coding best practices, the MANUFACTURER can decide to avoid using these or more functions.

   NOTE 3 Information on bad practices are available from coding standards, library providers, tools and other sources.

c) automated tool use and settings (for example, for static analysis tools),

d) general secure coding best practices,

   NOTE 4 The secure coding best practices can be based on published specifications, for example ISO/IEC TR 24772, MISRA-C or SEI CERT C and SEI CERT C++ coding standards.

e) validity checking of all inputs that cross a TRUST BOUNDARY,

f) error handling.

The MANUFACTURER should evaluate each type of alert from static analysis whether it justifies a code change.

The application of secure coding standards can be based on SECURITY ARCHITECTURE, programming technology and context.

Annex B
(informative)

Guidance on implementation of SECURITY LIFE CYCLE ACTIVITIES

B.1 Overview

CYBERSECURITY of a PRODUCT containing software can be supported by SECURITY CAPABILITIES of that software – typically implementing protection from, detection of, response to and recovery from incidents that can compromise the CONFIDENTIALITY, INTEGRITY or AVAILABILITY of the PRODUCT'S ASSETS.

B.2 Related work

Although this document focuses on software there are additional SECURITY considerations for the physical device that the software is running on that should be included in all PROCESS ACTIVITIES. Examples are to reduce physical interface ports, like JTAG or unused USB ports, similar to limiting open network ports at the software level. Similarly, there are mitigations provided by the device, such as physical locks to provide access control to internal media.

The technical reports IEC TR 60601-4-5 and IEC TR 80001-2-2 give guidance for the identification and communication of such SECURITY CAPABILITIES. While these technical reports address medical devices, their concepts and measures can easily be transferred to HEALTH SOFTWARE.

Another aspect is related to the LIFE CYCLE: MANUFACTURERS of HEALTH SOFTWARE can establish PROCESSES that avoid or mitigate VULNERABILITIES or reduce their impact to the PRODUCTS' INTENDED PURPOSE. Some PROCESSES – for instance requirements engineering and THREAT / RISK ANALYSIS (TRA) link the perspective of PRODUCT aspects with the view on PROCESSES. It is important to understand that only the combination of both PRODUCT capabilities as well as measures in the LIFE CYCLE PROCESSES can provide effective CYBERSECURITY.

B.3 THREAT / RISK ANALYSIS

SECURITY incidents can affect the PRODUCT'S SAFETY or effectiveness. The specific relationship between VULNERABILITIES and risks regarding SAFETY or effectiveness depends on the design, implementation and purpose of the respective PRODUCT. A PRODUCT risk analysis for SAFETY therefore shall consider the effects of VULNERABILITIES to the key functions of the PRODUCT. As a part of that ACTIVITY (or ACTIVITIES), TRA is performed for the PRODUCT.

SAFETY is defined as freedom from unacceptable risk, where risk is the combination of severity and probability of potential harm. The harm is expressed as injury, damage to health, property or environment (see ISO/IEC Guide 51). Where the INTENDED USE is known, the impact of SECURITY incidents finally can be expressed in terms of severity of the respective harm. In this case, SECURITY RISK MANAGEMENT can be integrated in a general RISK MANAGEMENT as applied by the MANUFACTURER based on ISO/IEC Guide 51 or ISO 14971 for medical devices. When following such an integrated approach, it shall be considered that management of risks arising from unauthorized activities (i.e. SECURITY-related risks) requires the application of specific methods and techniques differing from those for risks arising e.g. from non-reliable software, electrical failures, radiation, biological contamination or use errors. These SECURITY-specific methods and techniques include TRA and others as described in this document. TRA aims at identifying and evaluating scenarios of intrusion and the resulting WEAKNESSES. The scenarios considered during TRA are based on the actual context of use, which is not limited to the PRODUCT'S INTENDED USE, however TRA takes the USE ENVIRONMENT into account. Those scenarios with an attacker exploiting a known VULNERABILITY can be considered as "foreseeable" with respect to PRODUCT RISK MANAGEMENT and are also part of the actual context of use.

NOTE 1 The INTENDED USE can typically be determined at PRODUCT level.

NOTE 2 ISO 14971 specifies the consideration of reasonably foreseeable misuse.

In case the USE ENVIRONMENT or other mitigation controls might fail to prevent a certain type of ATTACK, that scenario becomes “foreseeable” from the MANUFACTURER’S perspective. TRA identifies and evaluates such THREAT scenarios – taking into account:

a) the dedicated hardware with which the HEALTH SOFTWARE is intended to be use,
b) the intended operational context, and
c) the potential data/control flows from external actors into the HEALTH SOFTWARE.

B.4 THREAT and RISK MANAGEMENT

One outcome of applying THREAT and RISK MANAGEMENT is an evaluation of known VULNERABILITIES that can affect the HEALTH SOFTWARE’S ASSETS (data, software functions, software services) with respect to CONFIDENTIALITY, INTEGRITY or AVAILABILITY – and how that is related to the overall SAFETY, SECURITY and effectiveness of the PRODUCT as a whole.

Options for controlling SECURITY risk with remaining VULNERABILITIES include one or more of the following:

a) fixing the issue through one or more of the following:
   1) DEFENSE-IN-DEPTH strategy or design change;
   2) addition of one or more SECURITY requirements and/or capabilities;
   3) use of compensating mechanisms; and/or
   4) disabling or removing features; with respect to the safe and effective use of HEALTH SOFTWARE;
b) creating a remediation plan to fix the problem;
c) deferring the problem for future resolution (reapply this requirement at some time in the future) and specifying the reason(s) and associated risk(s); and
d) not fixing the problem, if the RESIDUAL RISK meets the acceptance criteria.

When the resolution decision is to fix the SECURITY-related issue in the PRODUCT implementation, the timing of the release of the fix can result in a SECURITY update to be deferred until the next release.

B.5 Software development planning

B.5.1 Development

B.5.1.1 Software development PROCESS

An appropriate development PROCESS for HEALTH SOFTWARE should implement a development/ maintenance/ support PROCESS as required in IEC 62304[8] and should additionally implement items of the list specified in 5.1.1.

B.5.1.2 Development environment SECURITY

HEALTH SOFTWARE shall be protected from any compromises via the development environment. For instance, the introduction of malicious software or the theft of credentials such as software signing certificates.

B.5.2 HEALTH SOFTWARE requirements analysis

B.5.2.1 HEALTH SOFTWARE SECURITY requirements

In some circumstances a system at a higher level has already defined a SECURITY level for this (sub)system. This is described per IEC 62443-3-2 in general and via IEC TR 60601-4-5 for Programmable Electrical Medical Systems (PEMS).

B.5.2.2 SECURITY requirements review

The implementation of SECURITY CAPABILITIES can have an impact on the PRODUCT'S SAFETY or effectiveness. This review can determine an appropriate requirement for implementing SECURITY CAPABILITIES in a balanced way.

B.5.3 Software architectural design

B.5.3.1 DEFENSE-IN-DEPTH ARCHITECTURE /design

DEFENSE-IN-DEPTH is an approach to CYBERSECURITY in which a series of defensive mechanisms are layered in order to protect information ASSETS. If one mechanism fails, another layer will thwart an ATTACK. This multi-layered approach with intentional redundancies increases the SECURITY of a system as a whole and addresses many different ATTACK vectors. DEFENSE-IN-DEPTH is commonly referred to as the "castle approach" because it mirrors the layered defenses of a medieval castle.

DEFENSE-IN-DEPTH reduces the likelihood of ATTACKS to succeed, it reduces the impact of ATTACKS and allows the target system to take compensating actions.

B.5.3.2 Secure design principles

The principles described in this requirement are relevant to the design of any system, whether for apps, client or server, cloud-based services, or Internet-of-Things devices. The specifics of their application will vary – a cloud service can require multiple administrative roles, each with its own least privilege, while an IoT device will require special considerations of the need for SECURITY updates and of the need to fail securely and safely.

However, the principles are general and provide valuable SECURITY guidance for the designers and architects of all classes of systems. Elements of such a PROCESS need additional specifications that depend on the programming environment and the information technology used. There are specifications from Standard-Developing Organisations (SDOs) or associations with more detailed specifications (potentially depending on technology or context).

B.5.3.3 SECURITY architectural design review

The ability of an ARCHITECTURE to ensure stable and predictable behavior is important, because adverse conditions can come intentionally or unintentionally; they can show up via adverse calls / data when HEALTH SOFTWARE is being used in its USE ENVIRONMENT.

B.5.4 Software unit implementation and VERIFICATION

Secure coding standards should incorporate the following principles:

- establish coding standards and conventions;
- use safe functions only (i.e. reliable functions);
- use current compiler and toolchain versions and secure compiler options;
- handle input and other data safely (i.e. in a restrictive, cautious way…);
- use static code analysis tools to find SECURITY issues early;
- handle errors.

NOTE 1 Best coding practices avoid coding based on unspecified or undefined behaviour of the programming environment.

NOTE 2 Static Code Analysis (SCA) detects the potential for errors such as buffer overflows, null pointer dereferencing, and similar.

NOTE 3 SCA can be done using a tool if one is available for the language used. In addition, static code analysis can be done on all source code changes including new source code.

B.5.5 Secure implementation

The MANUFACTURER can implement an ARCHITECTURE and design that allow for updating or substituting hardware components and SOFTWARE ITEMS – for example cryptographic modules. The goal here is to implement with technology agility in mind: e.g. encryption algorithms might potentially be broken at any time, even if they are considered current best practices, and encryption libraries can have VULNERABILITIES that undermine otherwise sound algorithms. In the example, a secure implementation should ensure that some encryption strategy specifies how applications and services should implement their encryption to enable transition to new cryptographic mechanisms, libraries and keys when the need arises. The above is just an example; substitution in the ARCHITECTURE also serves as a means to be able to support necessary SECURITY updates and upgrades.

B.5.6 Not used

B.5.7 Software system testing

B.5.7.1 SECURITY requirements testing

Subclause 5.7 on software system testing provides the requirements and more detail related to SECURITY testing.

An overview of some automated and manual testing techniques includes the techniques described from B.5.7.2 to B.5.7.5.

B.5.7.2 THREAT mitigation testing

As an example for THREAT mitigation testing, input VALIDATION testing plays an important role.

Input VALIDATION testing tries to detect undesired system behavior when incorrect data or excessive load of data are sent to a system interface. Often automated tools are used and the more specialized the tool is for a certain interface protocol, the more accurate the test results will be. Examples are fuzz testing, buffer overflow and format error testing. Specialized injection testing techniques exist for protocols such as SQL, LDAP, XML and cross-site scripting.

B.5.7.3 VULNERABILITY scanning

VULNERABILITY scanning is the automated detection of known VULNERABILITIES. Scanners will detect installed software, open network ports, operating system configuration and other SECURITY relevant information. Many VULNERABILITY scanners allow for both authenticated and unauthenticated scans. An authenticated scan means that the tool has administrative system credentials to bypass certain protections and will be able to assess the systems configuration with much more detail and accuracy. OWASP (“the Open Web Application SECURITY Project” foundation) maintains a list of VULNERABILITY scanning tools.

B.5.7.4 Penetration testing

Penetration testing, also called pen-testing, focuses specifically on compromising CONFIDENTIALITY, INTEGRITY or AVAILABILITY. It can involve defeating multiple aspects of the DEFENSE-IN-DEPTH design. For example, bypassing authentication to access the PRODUCT, using elevation of privilege to gain administrative access and then compromising CONFIDENTIALITY by breaking encryption. As this example shows, penetration testing involves approaching testing like an attacker and often involves exploiting chained VULNERABILITIES in a PRODUCT using both tools and manual skills. Results of the VULNERABILITY scanning and other tests could provide valuable input to develop manual ATTACK scenarios.

Penetration testing should include an individual who was not involved in the development of the HEALTH SOFTWARE.

B.5.7.5 Managing conflicts of interest between testers and developers

Objectivity aims at making decisions by applying established methods to facts, such that any other tester can reproduce the decision at a later time. Independence can support objectivity. The MANUFACTURER can implement a (semi-)independent internal testing team and/or the use of a SECURITY test organization.

Static code analysis and SOFTWARE COMPOSITION ANALYSIS (binary analysis) typically depend on the use of automated tools. These tests are mentioned in IEC 62443-4-1 but that document does not specify independence requirements. The way how automated tools are being used and how their outcomes are being interpreted needs objectivity as well, however the calibration required for different tools (and different programming environments) does not ensure that outcomes are always consistent across tools. Therefore, implementing reproducibility within a given tool chain is recommended[41].

Annex C
(informative)

THREAT MODELLING

C.1 General

THREAT MODELLING is a systematic approach for analyzing the SECURITY of an item in a structural way such that VULNERABILITIES can be identified, enumerated, and prioritized, all from a hypothetical attacker's point of view. THREAT MODELLING can be applied to a wide range of things, including software, devices, systems, networks, distributed systems and business PROCESSES. THREAT MODELLING typically employs a systematic approach to identify ATTACK vectors and ASSETS most desired by different THREAT actors. This leads to a decomposition of the item (software, device, system, and so on) to look at each possible ATTACK vector and ASSET individually and determine to which kind of ATTACKS they are vulnerable. From this, a list of VULNERABILITIES can be created and ordered in terms of risk, potential to impact SAFETY, effectiveness, or any other criteria deemed appropriate (like privacy).

There are various approaches to creating a THREAT MODEL that range from making a list of known VULNERABILITIES to adopting a framework; some examples are described from C.2 to C.10.

A THREAT actor or malicious actor is a person, group, or organization attacking an organization with the potential to impact, the SAFETY or SECURITY of systems. THREAT actors will have different motivations to ATTACK certain organizations or systems such as financial gain, data theft, intellectual property theft or just to disrupt the trust in the targeted organization. A THREAT actor can have limited skills and resources (script kiddy) or significant skills and resources (cyberterrorists and state actors). THREAT actors are often further categorized as intentional or unintentional (use errors) and can be external or internal to the organization. Typical THREAT actors that could be taken into consideration during THREAT MODELLING are insiders (clinical users, system administrators), script kiddies, cyber criminals, hacktivists, and nation state actors.

C.2 ATTACK-defense trees

An ATTACK-Defense Tree (ADTree) is a node-labeled rooted tree describing the measures an attacker might take to ATTACK a system and the defenses that a defender can employ to protect the system.

C.3 CAPEC / OWASP / SANS

A basic approach is to use lists of known top THREATS such as the OWASP Top 10 or the CWE/SANS Top 25. The "Common Attack Pattern Enumeration and Classification" (CAPEC) has a more comprehensive dictionary of known patterns of ATTACK employed by adversaries to exploit known WEAKNESSES.

C.4 CWSS

The Common WEAKNESS Scoring System (CWSS) both identifies VULNERABILITIES and provides a scoring system to prioritize them. It is a collaborative, community-based effort that focuses on analyzing software and reported bugs to determine the relative importance of the detected WEAKNESSES.

C.5 DREAD

DREAD is a classification scheme for quantifying, comparing and prioritizing the amount of risk presented by each evaluated THREAT. DREAD modelling focuses on risk rating. The DREAD algorithm is used to compute a risk value, which is an average of all five categories: **D**amage, **R**eproducibility, **E**xploitability, **A**ffected users, and **D**iscoverability.

C.6 List known potential VULNERABILITIES

One can attempt listing all the VULNERABILITIES that could affect your system. While it is impossible to list all potential VULNERABILITIES, one should concentrate on those VULNERABILITIES that could be exercised by known THREATS.

C.7 OCTAVE

OCTAVE is a heavyweight risk methodology approach originating from Carnegie Mellon University's Software Engineering Institute (SEI) in collaboration with CERT. OCTAVE focuses on organizational risk, not technical risk.

C.8 STRIDE

STRIDE is a model for system decomposition, by characterizing known THREATS according to the kinds of EXPLOITS used. The STRIDE acronym stands for each of the categories: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of Privilege. STRIDE does not include a scoring system.

C.9 Trike

Trike is a THREAT MODELLING framework with similarities to the STRIDE and DREAD THREAT MODELLING PROCESSES. Trike differs in that it uses a risk-based approach with distinct implementation, THREAT, and risk models, instead of using the STRIDE/DREAD aggregated THREAT MODEL (ATTACKS, THREATS, and WEAKNESSES).

C.10 VAST

VAST is an acronym for Visual, Agile, and Simple THREAT MODELLING. The principle of this approach is the necessity of scaling the THREAT MODELLING PROCESS across the infrastructure and entire software development LIFE CYCLE. The approach integrates into an agile software development methodology. The methodology provides an application and infrastructure visualization scheme such that the creation and use of THREAT MODELS do not require specific SECURITY subject matter expertise.

Annex D
(informative)

Relation to practices in IEC 62443-4-1:2018

D.1 IEC 81001-5-1 to IEC 62443-4-1:2018

IEC 81001-5-1	IEC 62443-4-1:2018		IEC 81001-5-1	IEC 62443-4-1:2018
4.1.1	Not in IEC 62443-4-1		5.7.4	SVV-4
4.1.2	SM-2		5.7.5	SVV-5
4.1.3	SM-3		5.8.1	SM-11
4.1.4	SM-4		5.8.2	SG-5, SG-6
4.1.5	SM-10		5.8.3	SM-6
4.1.6	SM-13		5.8.4	SM-8
4.1.7	DM-5		5.8.5	SM-11
4.1.8	DM-6		5.8.6	SM-12
4.1.9	SG-7		5.8.7	SG-4
4.2	Not in IEC 62443-4-1		6.1.1	SUM-5
5.1.1	SM-1, SM-5		6.2.1	DM-1
5.1.2	SM-7		6.2.2	SUM-1
5.1.3	SI-2		6.3.1	SUM-3
5.2.1	SR-3, SR-4		6.3.2	SUM-2, SUM-4
5.2.2	SR-5		6.3.3	SM-6
5.2.3	SM-9		7.1	SR-1
5.3.1	SD-2		7.2	SR-2
5.3.2	SD-4		7.3	Not in IEC 62443-4-1
5.3.3	SD-3		7.4	Not in IEC 62443-4-1
5.4.1	SD-4		7.5	Not in IEC 62443-4-1
5.4.2	SD-2		8	SM-1
5.4.3	SD-1		9.1	Not in IEC 62443-4-1
5.4.4	SD-3		9.2	DM-1
5.5.1	SI-2		9.3	DM-2
5.5.2	SI-1		9.4	DM-3
5.6	Not in IEC 62443-4-1		9.5	DM-4
5.7.1	SVV-1		A.4	SI-2
5.7.2	SVV-2		E.2	SG-1, SG-2, SG-3
5.7.3	SVV-3		E.3	SG-4