ETSI GS ISI 005 V1.1.1 (2015-11)

Information Security Indicators (ISI); Guidelines for security event detection testing and assessment of detection effectiveness

Disclaimer

This document has been produced and approved by the Information Security Indicators (ISI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/ISI-005

Keywords
ICT, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

List of figures

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Information Security Indicators (ISI).

The present document is included in a series of 6 ISI specifications.
These 6 specifications are the following (see figure 1 summarizing the various concepts involved in event detection and interactions between all specifications):

- ETSI GS ISI 001-1 [i.8] addressing (together with its associated guide ETSI GS ISI 001-2 [i.12]) information security indicators, meant to measure application and effectiveness of preventative measures,
- ETSI GS ISI 002 [i.9] addressing the underlying event classification model and the associated taxonomy,
- ETSI GS ISI 003 [i.11] addressing the key issue of assessing an organization's maturity level regarding overall event detection (technology/process/ people) in order to evaluate event detection results,
- ETSI GS ISI 004 [i.10] addressing demonstration through examples how to produce indicators and how to detect the related events with various means and methods (with a classification of the main categories of use cases/symptoms),
- **ETSI GS ISI 005** addressing ways to test the effectiveness of existing detection means within an organization, which is a more detailed and a more case by case approach than ISI 003 [i.11] one and which can therefore be complementary.

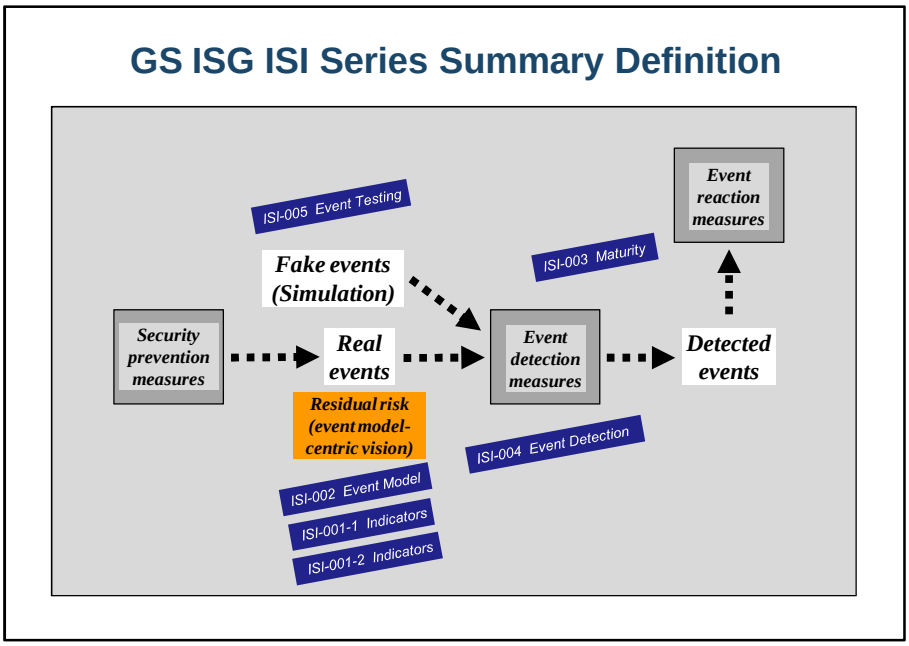

Figure 1: Positioning the 6 GS ISI against the 3 main security measures

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

The purpose of the present document is to describe strategies and techniques to test security event detection systems and to assess the effectiveness of such systems.

The present document also includes few examples of tests scenarios.

1 Scope

The present document provides an introduction and guidelines for the development of tests to check the capabilities of security event detection systems.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references.

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ISO 27004:2009: "Information technology - Security techniques - Information security management - Measurement".

[i.2] ISO/IEC/IEEE 29119-2: "Software and system engineering - Software Testing - Part 2 : Test process, 2013".

[i.3] IEEE 829™-2008: "Standard for Software and System Test Documentation".

[i.4] Recommendation ITU-T X.294: "OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications - Requirements on test laboratories and clients for the conformance assessment process".

[i.5] ISO/IEC 15408: "Information technology -- Security techniques -- Evaluation criteria for IT security".

[i.6] Common Weakness Enumeration (CWE).

NOTE: Available at https://cwe.mitre.org.

[i.7] Common Attack Pattern Enumeration and Classification (CAPEC).

NOTE: Available at https://capec.mitre.org.

[i.8] ETSI GS ISI 001-1: "Information Security Indicators (ISI); Indicators (INC); Part 1: A full set of operational indicators for organizations to use to benchmark their security posture".

[i.9] ETSI GS ISI 002: "Information Security Indicators (ISI); Event Model A security event classification model and taxonomy".

[i.10] ETSI GS ISI 004: "Information Security Indicators (ISI); Guidelines for event detection implementation".

[i.11] ETSI GS ISI 003: "Information Security Indicators (ISI); Key Performance Security Indicators (KPSI) to evaluate the maturity of security event detection".

[i.12] ETSI GS ISI 001-2: "Information Security Indicators (ISI); Indicators (INC); Part 2: Guide to select operational indicators based on the full set given in part 1".

[i.13] DIAMONDS project deliverables.

NOTE: http://www.itea2-diamonds.org/_docs/D3_WP4_T1_v1_0_FINAL_initial_test_patterns_catalogue.pdf.

[i.14] A. Vouffo Feudjio:"A Methodology For Pattern-Oriented Model-Driven Testing of Reactive Software Systems", PhD Thesis, February 2011.

NOTE: http://opus.kobv.de/tuberlin/volltexte/2011/3103/pdf/vouffofeudjio_alaingeorges.pdf.

[i.15] OWASP-AT-003: "Testing for Default or Guessable User Account".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI GS ISI 001-1 [i.8] and the following apply.

stimulation: single or sequence of activities in order to produce a security event: security incident (e.g. installation of an unauthorized application) or introduction of a vulnerability (e.g. misconfiguration of a critical device)

system under test: security event detection system or software to be tested

system under monitoring: system where the security event detection system is installed

test case: set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly

test pattern: expression of the essence of a well-understood solution to a recurring testing problem

test priority: level of (business) importance assigned to a test case

test selection: means of adapting Test Suites to the options supported by the Implementation and/or the priorities provided by the test developers, customer or other stakeholders or algorithms [i.4]

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AV	AntiVirus
CAPEC	Common Attack Pattern Enumeration and Classification (Mitre)
CCHIT	Certification Commission for Health Information Technology
CIA	Confidentiality Integrity Availability
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DNS	Directory Name Service
DOS	Denial of service
EICAR	European Expert Group for IT-Security
HTTP	Hyper Text Transfer Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IPS	Intrusion Prevention System
ISO	International Organization for Standardization

IT	Information Technology
NIST	National Institute of Standards and Technology (USA)
OS	Operating System
PC	Personal Computer
PIN	Persona Identification number
SFR	Security Functional Requirement
SIEM	Security Information and Event Management
SQL	Structured Query Language
SSL	Secure Socket Layer
SUT	System Under Test
TCP	Transport Control Protocol
UML	Unified Modelling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

4 Objectives of security event detection testing

4.1 Assessment of detection effectiveness

4.1.0 Introduction on assessment of detection effectiveness

The objective of testing security event detection systems is to be able to assess the effectiveness of the detection functionality. This evaluation can be performed in a laboratory (the detection system under test is not connected to an operational system) or in real operation (the detection system under test is connected to the real operational system).

Detection capability testing can be done before the deployment of the detection system. Test campaigns can also be organized regularly to assess the sustainability of the detection capability.

It should be noted that when detection capabilities are outsourced, specific audit clauses have to be defined in the contract.

The result of the assessment is not a single result but a set of both quantities and qualitative data.

4.1.1 Examples of quantitative results

4.1.1.1 Detection level

This measurement determines the rate of security events detected correctly by the SUT in a given environment during a particular time frame. The accuracy of that detection level is directly based on the sample of events used to perform the measurement. Due to the fact that as of today no standardized sample database exists, current published detection levels are not comparable between each other.

The other reason why results are not comparable is due to the fact that the detection level is directly linked to the detection rules configured in the tools (IDS, SIEM). The list of configured rules depends on each particular deployment and not on the installed tools.

4.1.1.2 Coverage of events specified in ETSI GS ISI 001-1

ETSI GS ISI 001-1 [i.8] specifies a list of events that may be detected in order to generate accurate indicators. The measurement of the detection level could be the amount (in percent) of security events that the system under test can detect compared to the list specified in the ETSI GS ISI 001-1 [i.8].

4.1.1.3 False-positive rate

This measurement determines the rate of false-positives produced by a detection system in a given environment during a particular time frame. A false-positive or false alarm is an alert caused by an event that is not a security event (vulnerability or incident).

4.1.2 Examples of qualitative results

Testing can also be used to characterize the type of detection implemented in the system under test. Detection type can be categorized in three main categories:

- Suspicious behaviours (exhibited either by targets or attackers) that deviate from usual and specified operations (also known in the literature as "anomaly detection").
- Exploitation underway of known software or configuration vulnerabilities (also known in the literature as "misuse detection").
- Other attacks requiring correlation (especially known structured and complex attack patterns).

Clause 6 of ETSI GS ISI 004 [i.10] provides more details on the technical characteristics of these three categories.

4.2 Conformity evaluation

For benchmarking or for procurement purpose, it could be necessary to evaluate the conformity of a security event detection system to its specifications. Specifications can include the list of security events that the system is able to detect or the list of sensors (collecting data) supported by the system.

If the detection system is limited to a product, the Common Criteria standard methodology [i.5] can be used to evaluate the conformity of the product to its specifications (its "security target" in the Common Criteria terminology).

4.3 Resistance to attacks

Another objective of the testing of a security event detection system could be to evaluate its resistance to attacks. To be efficient, the detection system should, either be unreachable by the attackers, or at least more resistant and resilient than the system under monitoring. It is therefore accurate to evaluate the resistance of the detection system to attacks.

The main objective of attacking a detection system is to deactivate detection capabilities. That objective can be reached:

1) by physical attacks on the equipment supporting the detection;
2) by software attacks on servers or probes.

If the detection system is limited to a product, the Common Criteria standard methodology [i.5] can also be used to evaluate the resistance of the product to attacks. The standard defines four levels of resistance:

1) the product is resistant to attacks performed by an attacker possessing Basic attack potential (AVA_VAN.1 & AVA_VAN.2 assurance requirements components [i.5]);
2) the product is resistant to attacks performed by an attacker possessing Enhanced-Basic attack potential (AVA_VAN.3 assurance requirements components [i.5]);
3) the product is resistant to attacks performed by an attacker possessing Moderate attack potential (AVA_VAN.4 assurance requirements components [i.5]);
4) the product is resistant to attacks performed by an attacker possessing High attack potential (AVA_VAN.5 assurance requirements components [i.5]).

5 Test framework

5.0 Introduction

This clause addresses general consideration for testing, i.e. test procedures, configurations that are needed to perform test campaign of detection systems. When applicable they have been derived and/or adopted from appropriate security testing activities [i.5].

5.1 Active vs. passive testing

Technically, the detection system should trigger over the presence of a certain type of security event. That event can be artificial (the tester generates the event, there in-after "Active testing") or the tester can wait for the occurrence of a real event (for example: the tester waits for the publication of a vulnerability in one of the deployed system, there-in-after "Passive testing").

The biggest benefit of the active testing is that it is not necessary to wait for the occurrence of real security events. Moreover, it could be impossible to test the detection of events having a very low probability of occurrence. The other difficulty is that the test should be aware of the occurrence of the security event even if it was not triggered by the SUT. It means that the tester needs another detection system with a better detection than the SUT in order to identify when the SUT does not detect what it should.

5.2 Active testing by stimulation

5.2.1 Objectives

As explained before, testing of security event detection capabilities is more accurate using active testing, when feasible. The objective for the tester is here to stimulate the detection procedures and mechanisms through the injection of events in the system under the monitoring of the detection system.

The tools and techniques used by the tester to stimulate the detection system are described in clause 6 of the present document.

5.2.2 Testing strategy

To make the interpretation of results easier, tests scenarios should be elaborated to trigger as much as possible a single security event detection. But to be representative of operational conditions, normal system activity should also be present.

While testing in the operational environment generates naturally such conditions, special activity generators should be developed for testing in labs.

Depending on the objective of the test campaign (detection effectiveness measurement, conformity evaluation, resistance to attacks), different testing strategies should be elaborated.

5.2.3 Stimulation location

5.2.3.0 Introduction on stimulation location

A tester can stimulate the SUT in two different ways: by the creation of the event or by the creation of the effects of the event. In addition, the tester should create « noise » simulating normal system usage in order to verify if the detection system is able to extract accurate events symptoms in that noise.

5.2.3.1 Noise generation

For testing in a laboratory, the tester needs to generate activity in the system under monitoring. For testing in real operational systems, the tester needs to evaluate if the current activity is sufficient or if he needs to stress the system with additional activity.

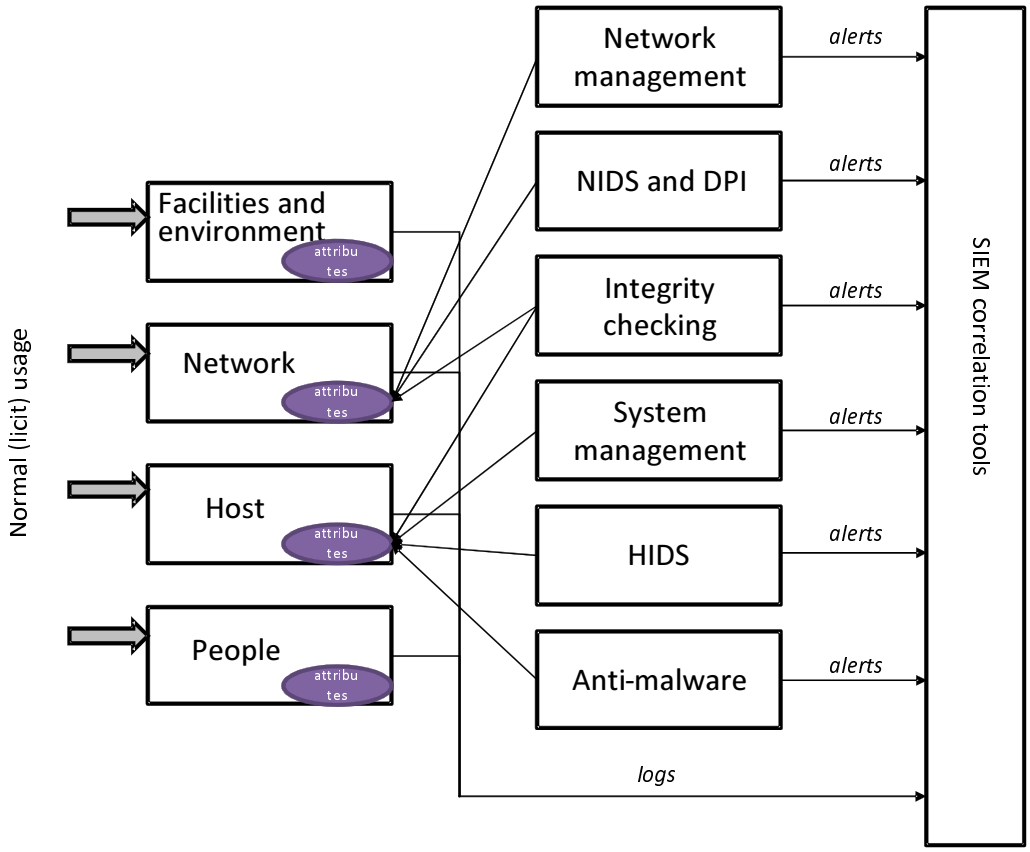

Figure 2: Noise generation

If the SUT generates alerts during this step, it can be considered as a false-positive.

5.2.3.2 Generation of events

The tester generates the event (or a suite of events) that has to trigger the SUT.

In practice, the tester performs the "action" ("what") on the "target" that characterize the tested security event.

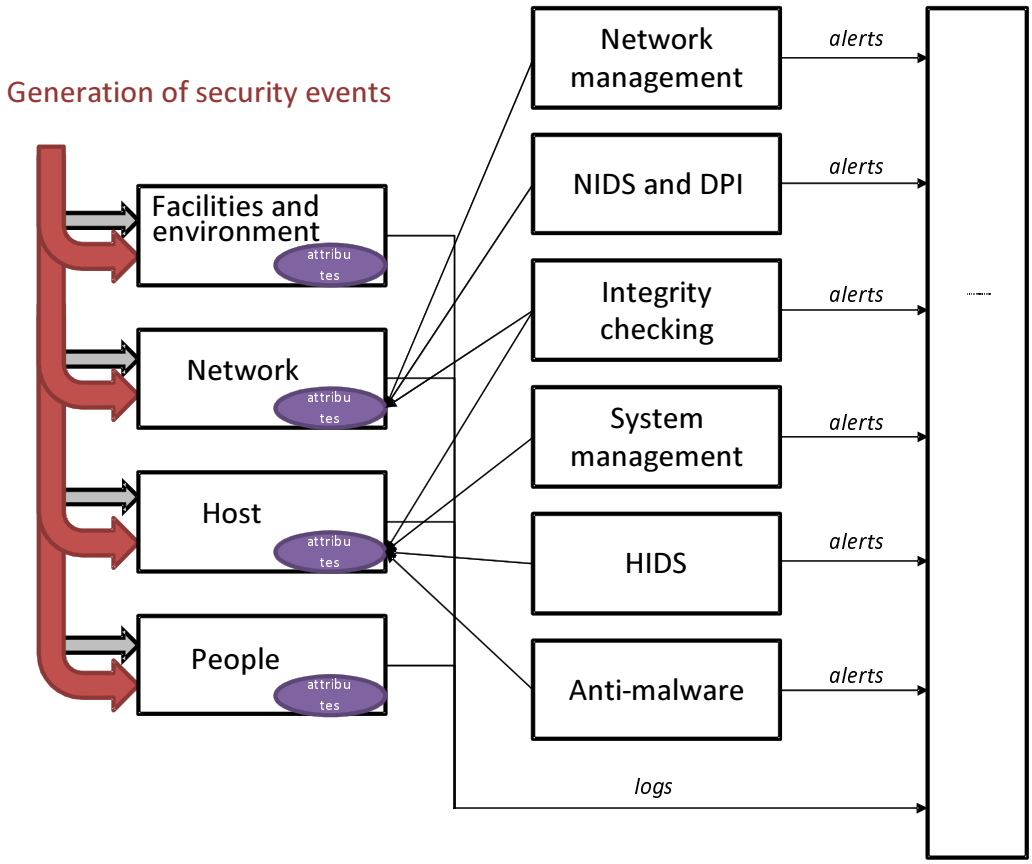

Figure 3: Generation of events

5.2.3.3 Generation of the event effects

The tester creates in the system the expected effects of the simulated security event that has to trigger the SUT. For example, the tester modifies the content of a file, stops a process, switches of an equipment or forges a false log event. These actions should trigger the sensors or the correlation of the detection system.

As defined in ISO 27004 [i.1], attributes are property or characteristic of an object.

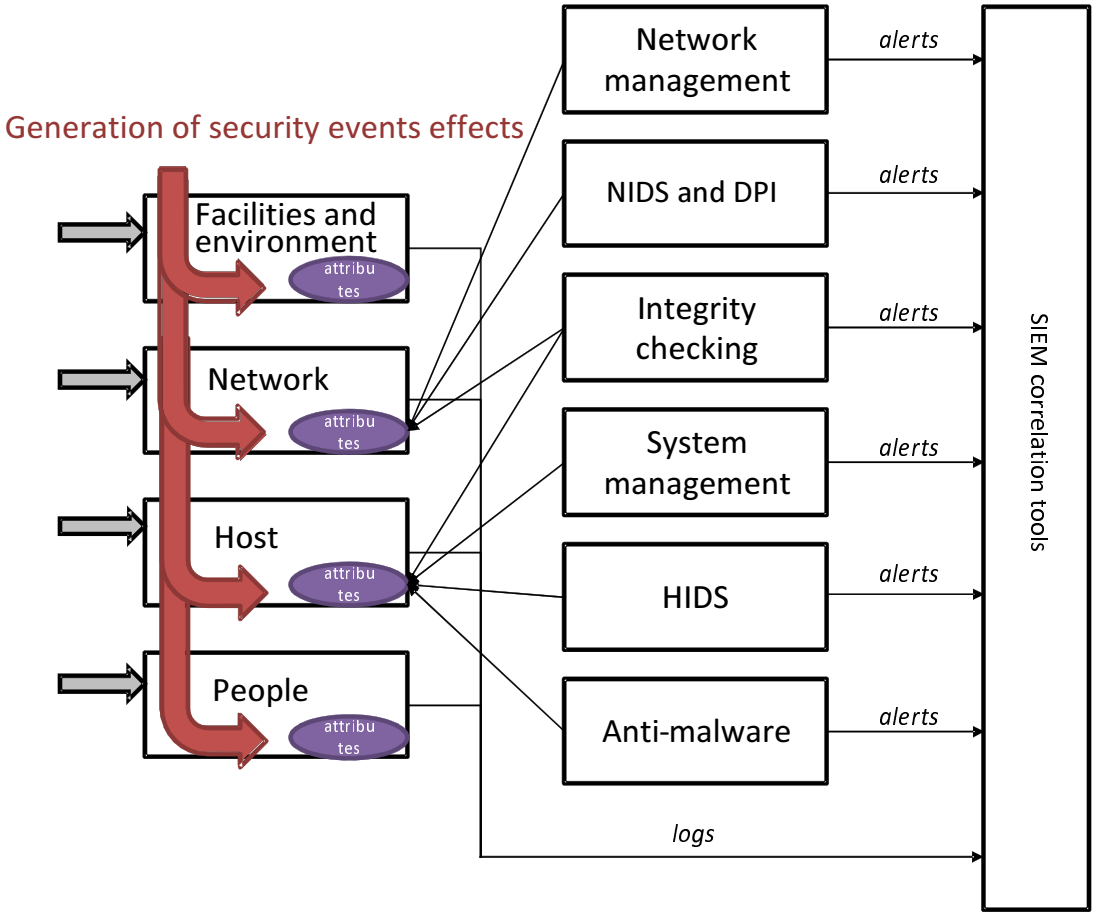

Figure 4: Generation of the event effects

5.2.3.4 Generation of alerts

When the attack or the modification of the operational system is difficult, a solution is to inject alerts into the detection system.

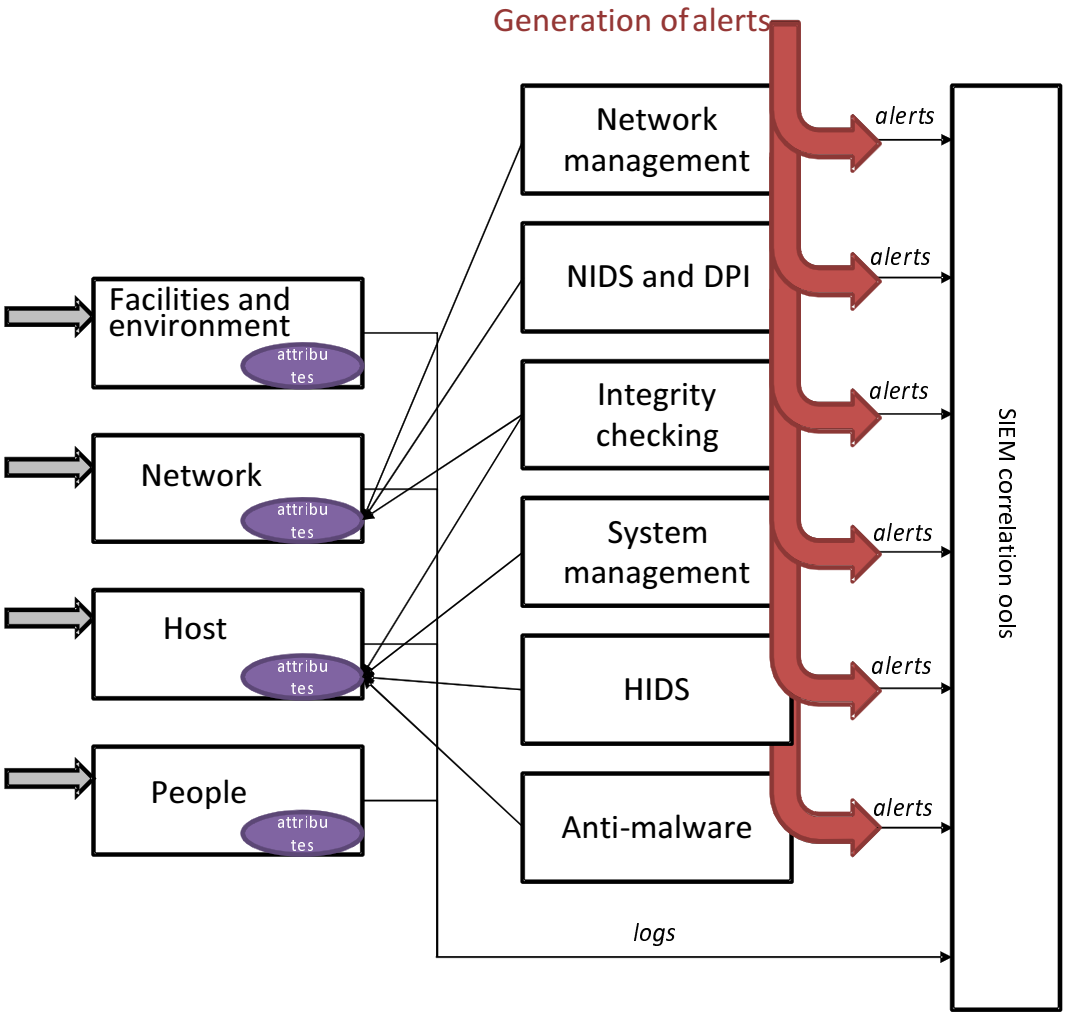

Figure 5: Generation of alerts

5.3 Test methodology

5.3.0 Introduction on test methodology

The test methodology is related to the generic security testing methodology, that is characterized by the following main steps:

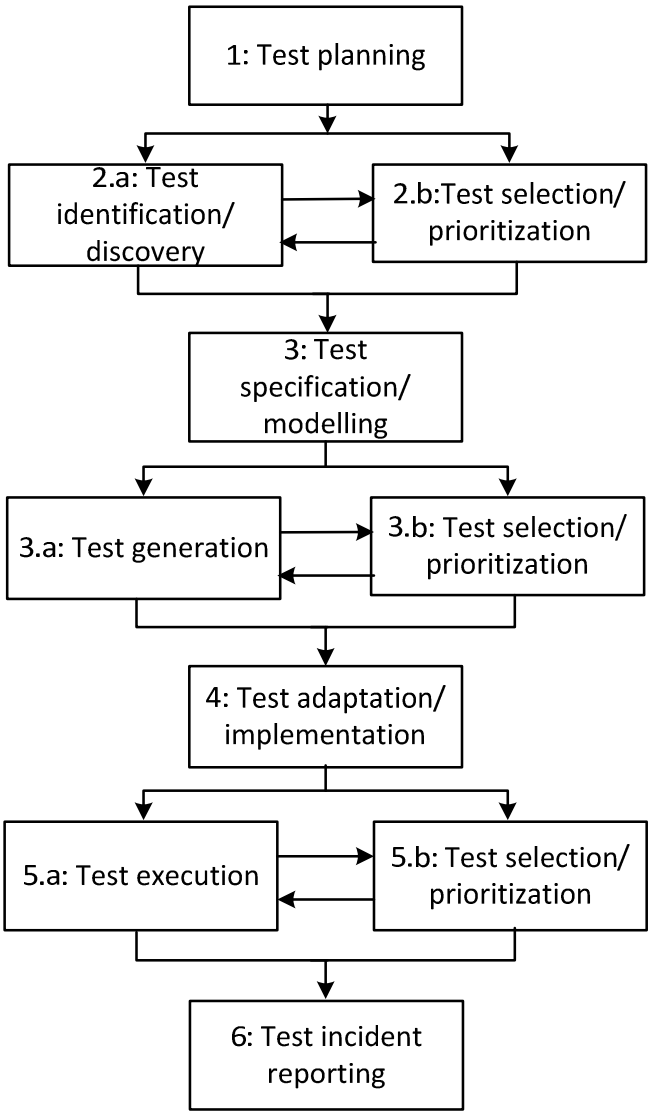

Figure 6: Test process

In the following, the steps of the process are described in more detail

Step 1: Test planning

The test planning is the activity of developing the test plan. Depending on where in the project this process is implemented this may be a project test plan or a test plan for a specific phase, such as a system test plan, or a test plan for a specific type of testing, such as a performance test plan (adapted from ISO/IEC/IEEE 29119-2 [i.2]).

Step 2.a: Test identification/discovery

Test identification/discovery is the activity of identifying/discovering test scenarios or areas or vulnerabilities in the systems where the testing should be focused. The discovery activity may be performed by e.g. use of network discovering techniques, vulnerabilities scanners, or through risk assessment.

Step 2.b: Test selection/prioritization

The activity of prioritizing and selecting potential test scenarios that are identified in step 2.a.

Step 3: Test specification/modelling

The activity of defining the model for test generation (i.e. computer-readable behavioural model that describes the intended external operational characteristics of a system, i.e. how the system being modelled interacts with its environment, in terms of the system interface) form which tests will be generated.

Step 3.a: Test generation

The automatic derivation of abstract test cases in one or more different formats from a model based on user defined test selection criteria.

Step 3.b: Test selection/prioritization

The process or the result of choosing a subset of tests during test generation from a larger or infinite set of tests which can be derived from a model.

Step 4: Test adaptation/implementation

The process of making the abstract test cases that are generated from the test models into concrete tests that can be executed.

Step 5.a: Test execution

The test execution is the process of running the test procedure resulting from the test design and implementation process on the test environment established by the test environment set-up and maintenance process. The test execution process may need to be performed a number of times as all the available test procedures may not be executed in a single iteration (adapted from ISO/IEC/IEEE 29119-2 [i.2]).

Step 5.b: Test selection/prioritization

The activity of prioritizing and selecting tests to be executed. The selection criteria may e.g. be based on a risk assessment. The activity may also involve mutation/fuzzing of concrete executable test cases.

Step 6: Test incident reporting

The test incident reporting is the process of managing the test incidents. This process will be entered as a result of the identification of test failures, instances where something unusual or unexpected occurred during test execution, or when a retest passes (adapted from ISO/IEC/IEEE 29119-2 [i.2]). In test-based risk assessment, the incident reporting activity may involve an assessment of how the test results impact the risk picture.

5.3.1 Test planning

Test planning follows best practises e.g. standardized guidelines as defined in IEEE 829™-2008 [i.3]. Such guidelines apply to all software-based systems and support acquisition, supply, development, operation, and maintenance. In particular it defines the test tasks, required inputs, and required outputs. Furthermore, it proposes the contents of the test plans as well as the contents of the related test documentation.

5.3.2 Test identification

If applicable test selection/prioritization steps may benefit from related security indicator characteristics such as detectability, severity, frequency, and/or reliability as defined in ETSI GS ISI 001-1 [i.8].

A key choice for test patterns consists in stimulating detection capabilities by major events of ETSI GS ISI 001-1 [i.8] security indicators. The reason for this choice is to use fake events defined at the right level, i.e. not specific system dependent and moreover consistent with the whole ETSI ISG ISI series. Proceeding this way enables to embrace a large scope of systems and more easily compare various detection capabilities. The consequence however is to work out relevant test patterns as regards their universal nature (see clause 7). Thus, the stake is to work out methods to simulate events which are close to the reality regarding attacks or deviant behaviours. The 3rd field "How" in the ETSI GS ISI 002 [i.9] incident taxonomy together with the related field dictionary make up some useful inputs to help here.

In addition test events that are not used for indicator generation could be of interest in the context and for "calibration" of measurement instruments.

5.3.3 Test specification

The ETSI GS ISI 002 [i.9] security event classification and taxonomy allows the specifications of the characteristics of each security event. The main interest is that tests patterns can be developed for each of these characteristics and then combined to specify tests cases for a specific security event.

See clauses 7 and 8 for examples.

5.3.4 Test generation

Test generation is characterized by synthesis of test scenarios in the context of well-defined test architectures. Usually test methodologies distinguish different testing levels: component, integration, system and acceptance tests. Testing of security event detection capabilities can be understood as a kind of system and/or acceptance testing for ISI detection systems.

Basic test scenarios may be reused from well-known databases or developed in the context of specific security events and vulnerabilities.

5.3.5 Test adaptation

The final test scenarios to be executed will be derived from abstract test scenarios that have been developed during test generation. Test pattern need to be selected, completed, parameterized and prepared for particular systems under test and/or systems under monitoring.

5.3.6 Test execution

In principle test execution will be performed automatically in order to fulfil e.g. load or time constraints, unique conditions, and to support error free repeatability and test log production. Since some tests may require longer test runs some test harness may also be involved to produce simulated events or incidents.

Test execution may also require various auxiliary utilities, e.g. with forensics capabilities.

5.3.7 Test results analysis

Test results analysis focuses on the interpretation of the test logs, traces and coverage of planned test runs and variants. Again, automation is required to achieve efficiency and effectivity.

5.4 Tests side-effects

5.4.0 Introduction on tests side-effects

Detection systems testing can have side effects that need to be known and approved. Some of them could lead to legal issues for the sponsor of the test campaign and for the organization performing the tests.

5.4.1 Production disturbance

The objective of certain tests is to generate denial of services. If tests are performed on the operational system, it will impact operational services. It is advised to perform such kind of tests on preproduction systems. Unfortunately, unexpected denial of services also occur during other type of tests. For example, some aggressive network scans could cause the unavailability of a server or some web penetration tests could cause the corruption and then unavailability of database.

5.4.2 Access to personal data

The objective of certain tests is to access personal data. If the test is performed in an operational system, the tester will have access to real personal data. This item should be covered by contractual service agreements before performing the tests. That contract should describe the data protection means used by the tester during the test and in the report and should mandate the deletion of the recovered data stored by the tester.

5.4.3 Unwanted personal stress

The occurrence of a security incident is a stressful situation for IT staff and employees. Because certain tests need to be performed without informing involved people, the incident could create panic and unexpected reactions. Tests simulating the deletion of user data or the modification of public information should be performed carefully. For such kind of tests, it is advised to inform users before the incident simulation or when the detection system triggers in order to avoid potential crisis cells.

5.5 Summary of the methodology for the generation of test scenarios

The methodology for the generation of test scenarios using test patterns can be summarized as follows (figure 7).

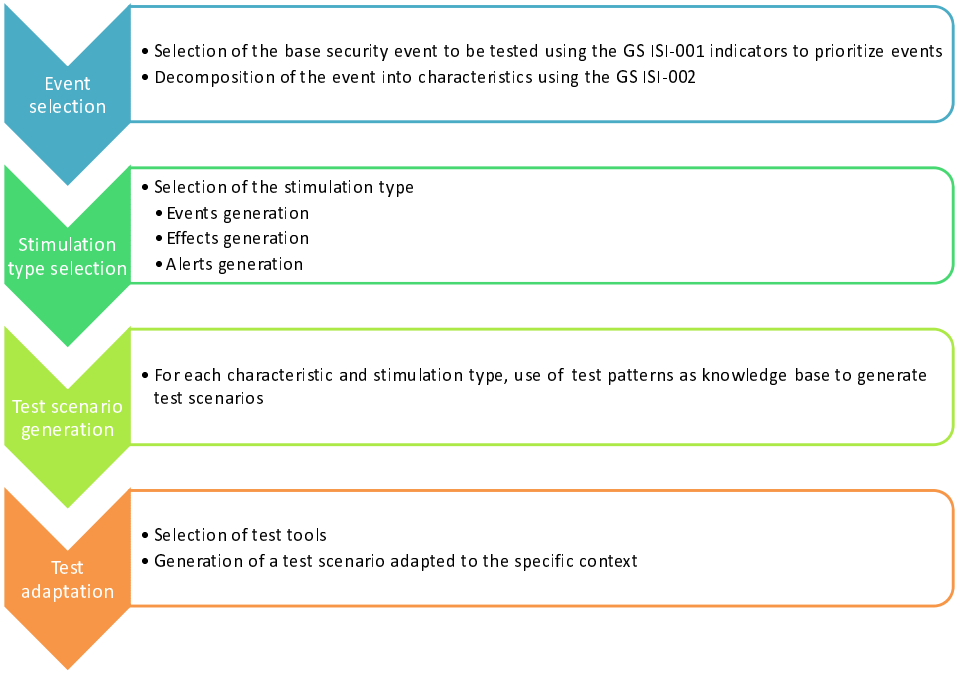

Figure 7: Summary of the methodology for the generation of test scenarios

6 Instruments for stimulation (tools & techniques)

6.1 Penetration testing

The objective of penetration testing is to carry out actions that a real attacker would perform. The tester defines the profile of the attacker that he needs to impersonate and the target that he needs to achieve.

In a standard penetration testing, the tester is usually free to perform the actions necessary to achieve his goal. In the context of the testing of detection system, it is recommended to agree with the tester about the techniques that he will use because it should determine the security event to be tested.

Using the taxonomy defined in ETSI GS ISI 002 [i.9], the penetration tester should act as the "who" and use the "how" to achieve the "what" on the "target".

Penetration testing can be done with or without the awareness of the IT operation staff.

6.2 Actions with internal participation

In some conditions, it could be easier to collaborate with the operator of the system under monitoring in order to generate the events or generate the effects of the events (see clause 5.2).

For the generation of the events, a user or an administration can act as a malicious actor. He can perform errors or illicit actions that should trigger the detection system. For example, and to reference the ETSI GS ISI 002 [i.9] "how" taxonomy, he can intentionally create an accidental modification of sensitive data, a configuration error, to send blackmail or to grant illicit access rights.

For the generation of the effects of the tested event, a user or an administrator can directly insert the consequence of the attack in the system. For example, and to reference the ETSI GS ISI 002 [i.9] "what" and "on what kind of assets" taxonomy, he can modify the content of a sensitive file, install a Trojan horse, a key logger or store inappropriate content in order to observe the reaction of the detection system.

The consequence of the actions should be assessed before being carried out because the actions could result in a dangerous posture for the system under monitoring (for example after the deactivation of a security function). Restoration procedures should also be defined before performing the actions.

Actions can be carried out with or without the awareness of the IT operation staff. A standard user can be the participant without the awareness of the administrators.

6.3 Known-vulnerable systems

The testing of detection system requires the implementation of real attacks (despite reducing as much as possible the impact on the operational system). First, finding exploitable vulnerabilities in a system is a huge task. Second, if the tester is able to find vulnerable operational systems, it becomes more urgent to fix the issue before a real malicious exploitation than to continue the testing.

A simple solution is then to introduce known-vulnerable systems. Deprecated versions of software are a good source to build known-vulnerable systems despite the fact that it is becoming increasingly difficult to download deprecated versions of product. Testers are advised to build a database of such known-vulnerable software with their associated exploits. For frequently used products, it is possible to find in vulnerabilities database a list of known vulnerabilities for each version of software and sometimes the associated exploit scripts allowing the exploitation of the vulnerability. The tester can then use that exploit to simulate a real attack.

In order to reduce unwanted side-effects such as the use of that known-vulnerable system by real attackers, it is highly recommended to isolate as much as possible that system from the other operational systems.

Another solution is not to use real known-vulnerable products but *Honeypot* tools instead. A *honeypot* is a trap used to simulate a real known-vulnerable system, while being actually isolated from the real operational system and being monitored. Honeypots are commonly used for malware and botnet detections but as seen here, they could be useful for detection testing. The tester attacks the honeypot and observes the detection system reaction. The benefit of using a honeypot and not a known-vulnerable system is that in case a real attacker attacks the honeypot, he does not impact real operational data and services.

6.4 Hacking tools

The tester needs to use real hacking tools if he wants to imitate the attacker's behaviour. ETSI GS ISI 004 [i.10] defines a 5-step attack stream:

1) Exploration and spying
2) Sabotage
3) Intrusion
4) Malware and utilities installation
5) Camouflage

Tools are available for all these attack steps. Vulnerability scanners can be used to simulate the exploration step. Traffic generators can be used to trigger saturation of control devices (log filling, connection tables filling, CPU saturation, etc.). Exploit scripts can be used for the intrusion step. For the utilities installation step, the development of a malware is not required; standard system utilities can be sufficient to open network connections, to send emails or to delete/modify files content. For the camouflage step, it is recommended to develop adhoc tools because the attacker needs to adapt the tool behaviour to the targeted system. The camouflage tools needs to be as similar as possible to a normal user behaviour.

The ownership and the use of attack tools can be prosecuted in few countries, even for testing purpose. Before each testing campaign, the tester should check local applicable laws on that subject.

7 Examples of detection tests

7.0 Detection tests specification

In order to help to the definition of test case, test patterns can be used as a knowledge base. A test pattern is a template that can be used to build a complete test scenario. Detection test pattern can be generated for each characteristic of the events as defined in ETSI GS ISI 002 [i.9].

Combining test patterns associated to each event characteristic allows to build an accurate and complete detection test scenario.

For example, a test pattern for the "Who"= "Malicious act - external agent" can be:

	For Event generation	For Effects generation	For alert generation
Prerequisites	Definition of the malicious external agent profile	Not necessary for effects generation, see other characteristics	Not necessary for alert generation, see other characteristics
Test scenario	The tester acts like the defined malicious external agent profile (list of actions depends on other characteristics) and observes the reaction of the SUT.	Not necessary for effects generation, see other characteristics	Not necessary for alert generation, see other characteristics
Potential side-effects	See other characteristics	Not necessary for effects generation, see other characteristics	Not necessary for alert generation, see other characteristics

For example, a test pattern for the "what" = "Installation of unauthorized software programs on a system (without the owner's consent)" can be:

	For Event generation	For Effects generation	For Alert generation
Prerequisites	Development of a software ("fake-malware") performing the illicit action defined in the "CIA consequences" characteristics	Development of a software ("fake-malware") performing the illicit action defined in the "CIA consequences" characteristics	Presence of an antimalware software in the targeted system Analysis of the format of alerts of the antimalware
Test scenario	The tester (by penetration testing or with an internal help) tries to install the "fake-malware" using the "how" technique and observes the reaction of the SUT.	A user having sufficient privileges on the target (see "on what asset" characteristics) installs the "fake-malware" and observes the reaction of the SUT.	If it is possible to inject fake-alerts into the system, the tester injects an alert declaring that a malware has been detected. If it is not possible to inject fake-alert, a user declares to the IT support desk that he detects a malware in his workstation.
Potential side-effects	Production disturbance	Production disturbance	-

For example, a test pattern for the "CIA consequences"= "Loss of integrity" can be:

	For Event generation	For Effects generation	For Alert generation
Prerequisites	Selection of the data to be modified	Selection of the data to be modified	Presence of an integrity checker in the targeted system Analysis of the format of alerts generated by the sensor
Test scenario	If the incident described by the "what" characteristics succeeds, the tester modifies the selected data.	A user having the sufficient privileges on the target (see "on what asset" characteristics) modifies the selected data and observes the reaction of the SUT.	If it is possible to inject fake-alerts into the system, the tester injects an alert declaring that a modification has been detected. If it is not possible to inject fake-alert, a user declares the support that he detects a modification of data.
Potential side-effects	Production disturbance Unwanted personal stress (in case if the modified data are personal data and if the owner is not aware of the test)	Production disturbance Unwanted personal stress (in case if the modified data are personal data and if the owner is not aware of the test)	-

The following examples of complete detection tests scenarios demonstration show the use of that detection test patterns. Examples are grouped by indicators extracted from ETSI GS ISI 001-1 [i.8].

7.1 IEX_INT.2: Intrusion on externally accessible servers

7.1.1 Base event

This indicator is generated from the detection of the following base event:

Base events
Detection of intrusion **Frequency:** Relatively high frequency **Severity:** 3 or 4 (depending on intrusion depth and according to successful access or not to personal data) **Detection means:** Automatic production possible (logs of server OS and/or of HTTP platforms and/or of Web applications, logs of IDS/IPS, and SIEM tool) **Detection level:** 1 (detection rate can be up to 15 %, very low rate demonstrated in the USA for thefts of credit card numbers - 50 % post-mortem rate after discoveries of fraud and intensive investigations)

7.1.2 Base event characteristics

The "Unauthorized access to a system and/or to information" technique category is decomposed into several sub-categories in ETSI GS ISI 002 [i.9]:

- Authentication attacks
- Use of a backdoor that has been installed during the software development stage or in production
- Various methods
- Technical methods for the 1st two kinds of events
- Other

The consequence is that the characteristics of the security events to be tested are the following:

Intrusion on externally accessible server						
Test reference	**Who**	**What**	**On what asset**	**How**	**Status**	**CIA consequences**
T.IEX_INT.2-1	Malicious act - external agent	Unauthorized access	Systems / Perimeter	Authentication attacks	Succeeded	Loss of integrity
T.IEX_INT.2-2	Malicious act - external agent	Unauthorized access	Systems / Perimeter	Use of a backdoor that has been installed during the software development stage or in production	Succeeded	Loss of integrity

7.1.3 Legitimate traffic

The tester should ensure that legitimate traffic is ongoing during the test campaign. Licit traffic for this test corresponds to connections of users to the perimeter system with, sometimes, errors when typing their credentials.

If the normal traffic is not sufficient in the operational system or in the lab, the tester can develop robots to simulate user's connections.

7.1.4 T.IEX_INT.2-1 testing

7.1.4.1 Stimulation type selection

The most accurate stimulation for that event is to generate the event and to observe the reaction of the SUT.

7.1.4.2 Test patterns selection

Using the specification of the characteristics of the base event, accurate test patterns can be selected.

Event to be simulated	**T.IEX_INT.2-1: Intrusion on externally accessible servers (authentication attack)**				
	Who	**What & On what asset**	**How**	**Status**	**CIA consequences**
	Malicious act - external agent	Unauthorized access / System-Perimeter	Authentication attacks	Succeeded	Loss of integrity and confidentiality
Prerequisites	Definition of the malicious external agent profile	Definition of the targeted system	Access to the authentication interface Availability of a brute-force attack tool for the authentication protocol	-	Selection of the data to be modified or to be accessed
Test scenario	The tester acts like the defined malicious external agent profile (list of actions depends on other characteristics) and observe the reaction of the SUT	See "how" characteristics	1. the tester tries easy, default credentials 2. the tester tries smarter attacks (e.g. replay, dictionary) 3. the tester tried brute-force attack on the authentication mechanism	See "CIA consequences "	If the incident described by the "what" characteristics succeeds, the tester modifies the selected data
Potential side-effects	See other characteristics	Access to sensitive data protected by the access control	Access to sensitive data (password of the attacked account)	See "CIA consequences	Production disturbance Unwanted personal stress (in case if the modified data are personal data and if the owner is not aware of the test) Access to personal data

7.1.4.3 Test adaptation

The tester defines the malicious external agent profile, the targeted system, and the data to be modified in case of success of the authentication attack.

Depending on the targeted system and the authentication protocol used in that system, the tester collects or develops tools to perform the scenario.

The tester should provide a detailed test case description for approval before performing the test.

7.1.5 T.IEX_INT.2-2 testing

7.1.5.1 Stimulation type selection

The most accurate stimulation for that event is to generate the event and to observe the reaction of the SUT.

7.1.5.2 Test patterns selection

Using the specification of the characteristics of the base event, accurate test patterns can be selected.

Event to be simulated	**T.IEX_INT.2-2: Intrusion on externally accessible servers (backdoor exploitation)**				
	Who	**What & On what asset**	**How**	**Status**	**CIA consequences**
	Malicious act - external agent	Unauthorized access / System-Perimeter	Backdoor exploitation	Succeeded	Loss of integrity and confidentiality
Prerequisites	Definition of the malicious external agent profile	Definition of the targeted system	Presence of a backdoor in the targeted system Availability of tools permitting to exploit the backdoor	-	Selection of the data to be modified or to be accessed
Test scenario	The tester acts like the defined malicious external agent profile (list of actions depends on other characteristics) and observe the reaction of the SUT	See "how" characteristics	The tester uses the exploitation tools to intrude the targeted system	See "CIA consequences"	If the incident described by the "what" characteristics succeeds, the tester modifies the selected data
Potential side-effects	See other characteristics	Access to sensitive data protected by access control mechanisms	Require the installation of a backdoor on the system that could be exploited by a real attacker	See "CIA consequences	Production disturbance Unwanted personal stress (in case if the modified data are personal data and if the owner is not aware of the test) Access to personal data

7.1.5.3 Test adaptation

The tester defines the malicious external agent profile, the targeted system, and the data to be modified in case of success of the attack.

If the operational targeted system does not contain any backdoor or vulnerability that can be exploited, it is better to change the stimulation type and to modify directly the targeted file rather than to introduce a vulnerability in an operational system only for the purpose of the test.

The tester should provide a detailed test case description for approval before performing the test.

7.2 IEX_DOS.1: Denial of service attacks on websites

7.2.1 Base event

This indicator is generated from the detection of the following base event:

Base events
Detection of an attack on a given website coming from the same origin within a limited continuous timeframe, and a significant incident defined as a user noticeable disturbance and performance drop in the website access. **Frequency:** Relatively high frequency, though very uneven over time **Severity:** 4 (if complete blockage of server or network) **Detection means:** Possible automatic production for DoS attacks (logs of databases and Web applications, system administration tools, and SIEM tool) and for DDoS attacks (network administration tools for perimeter areas) **Detection level:** 3 (detection rate can be up to 100 %)

7.2.2 Base event characteristics

DoS coming from the same origin						
Test reference	**Who**	**What**	**On what asset**	**How**	**Status**	**CIA consequences**
T.IEX_DOS.1-1	Malicious act - external agent	Information system remote disturbance	Systems / Perimeter	DoS methods	Succeeded	Loss of availability

7.2.3 Legitimate traffic

The tester should ensure that legitimate traffic is ongoing during the test campaign. Legitimate traffic for this test corresponds to connections of users to the perimeter system.

If the normal traffic is not sufficient in the operational system or in the lab, the tester can develop robots to simulate users' connections.

7.2.4 T.IEX_DOS.1-1 testing

7.2.4.1 Stimulation type selection

The most accurate stimulation for that event is to generate the event and to observe the reaction of the SUT.

If the targeted system integrates anti-DOS protection mechanisms, it could be difficult to perform the attack. On the other hand, to generate the effects (system not accessible) could also be unacceptable for business reasons.

Therefore, the best solution is to generate alerts into the detection system.

7.2.4.2 Test patterns selection

Using the specification of the characteristics of the base event, accurate test patterns can be selected.

Event to be simulated	T.IEX_DOS.1-1: DoS coming from the same origin				
	Who	**What & On what asset**	**How**	**Status**	**CIA consequences**
	Malicious act - external agent	Information system remote disturbance / System-Perimeter	DoS methods	Succeeded	Loss of availability
Prerequisites	Not necessary for alert generation	Definition of the targeted system	Not necessary for alert generation	-	Presence of an availability checker in the targeted system Analysis of the format of alerts generated by the sensor
Test scenario	Not necessary for alert generation	See "how" characteristics	Not necessary for alert generation	See "CIA consequences"	If it is possible to inject fake-alerts into the system, the tester injects an alert declaring that an unavailability has been detected
Potential side-effects	Not necessary for alert generation	System disturbance	Not necessary for alert generation	See "CIA consequences	Unwanted personal stress (in case if the unavailable service is critical and if the support is not aware of the test)

7.2.4.3 Test adaptation

The tester needs to analyse the format of the alerts generated by the availability checker installed in the targeted system. He also needs to investigate if it is possible to inject fake-alerts. If not, the tester needs to find the procedure for a user to declare the incident.

The tester should provide a detailed test case description for approval before performing the test.

7.3 IEX_MLW.3: Malware installed on workstations

7.3.1 Base event

This indicator is generated from the detection of the following base event:

Base events
Detection of a malware on workstations by non-conventional means (other than AV and standard IPS) **Frequency:** Relatively high frequency **Severity:** 2 to 4 (depending on the level of increase of the system load of PCs, or depending on the existence or absence of Trojan horses or bots) **Detection means:** Possible automatic production (detection by monitoring unusual system loads - typically increase after PCs are put to sleep, and/or by means of suspicious outgoing HTTP links to proxies - case of Trojan horses or bots, and/or by IDS at outbound network perimeter, and/or by users. PC system administration tools and/or logs of proxies and/or of firewalls, and SIEM tool) **Detection level:** From 1 to 3 (depending on type and stealth of malware - detection of Trojan horses and bots virtually impossible without SIEM tools, with the latter case providing detection rates possibly attaining 50 % for the best ones, but detection rate most often much lower and even non-existent, notably for the most sophisticated state-sponsored attacks)

7.3.2 Base event characteristics

Malware installed on workstations						
Test reference	**Who**	**What**	**On what asset**	**How**	**Status**	**CIA consequences**
T.IEX_MLW.3-1	All	Installation of unauthorized software programs on a system (without the owner's consent)	End-user devices / Multipurpose workstations	All	Succeeded	All

The specification of the "who", of the "how" and of the "CIA consequences" are not fundamental here because it is assumed that the installation of the malware has succeeded and the testing will focus on the detection after installation regardless of who did it and how he did it.

7.3.3 Legitimate traffic

The tester should ensure that legitimate traffic is ongoing during the test campaign. Legitimate traffic for this test corresponds to normal usage of the workstation but also normal system administration tasks like installation of software updates.

7.3.4 T.IEX_MLW.3-1 testing

7.3.4.1 Stimulation type selection

The most accurate stimulation for that event is to generate the effect of the event, i.e. to install a malware in the workstation and to observe the reaction of the SUT.

7.3.4.2 Test patterns selection

Using the specification of the characteristics of the base event, accurate test patterns can be selected.

Event to be simulated	**T.IEX_DOS.1-1: DoS coming from the same origin**				
	Who	**What & On what asset**	**How**	**Status**	**CIA consequences**
	All	Installation of unauthorized software programs on a system / End-user devices / Multipurpose workstations	All	Succeeded	All
Prerequisites	-	Development of a software not detected as unauthorized by antivirus or IDS ("fake-malware") performing an illicit action (data alteration, data leak, service deactivation)	-	-	-
Test scenario	-	Installation of the "fake-malware" on the workstation (without the users awareness or without the administrators awareness)	-	See "CIA consequences"	-
Potential side-effects	-	The potential side-effects depends on the illicit action coded in the "fake-malware"	-	See "CIA consequences	-

7.3.4.3 Test adaptation

The main task here is to develop the fake-malware adapted to the type of device selected. If a standard user can install it, the best strategy is to install the malware without the awareness of the administrators. If it is not possible to install the malware without system privileges, the device administrators should be involved in the test.

The tester should edit a detailed test case description for approval before to perform the test.

8 Examples of vulnerability tests

8.0 Introduction

Vulnerability tests can be used for three purposes:

1) To check if vulnerabilities detected by tests have also been detected by the detection system.

2) To check if the detection system is able to detect the test campaign, (vulnerability tests are close to attackers behaviour).

3) To evaluate the resistance of the detection system to attacks.

There are different approaches to vulnerability security test patterns:

- Abstract vulnerability test patterns
- Reuse of vulnerability test patterns
- Generic vulnerability test patterns (best suited to ETSI ISG ISI series issue)
- Vulnerability test patterns (also best suited to ETSI ISG ISI series issue)

As already indicated in the previous clause test patterns are related and have a strong link with the taxonomy of security incidents provided in ETSI GS ISI 002 [i.9].

NOTE: Any risks are given with the concrete test pattern (e.g. crash of server, connections, other resources, etc.).

8.1 Abstract vulnerability test patterns

Model-based testing approaches may allow the derivation of some test scenarios (i.e. pattern) from (semi-)formation system models by application of derivation algorithms that create test behaviour from use cases. Elements of test pattern include e.g. configuration, test scenario (conditions, body, optional postamble), test result expectation.

Setup of such scenarios require some suitable abstraction/convention for the scenarios definition (using a platform independent notation) and secondly the enrichment (instantiation and parameterization) of the tests for their execution.

8.2 Use of vulnerability test patterns from existing vulnerability test methods

Target detection systems under test can be understood as parts of security software products to be subject of security evaluations. Therefore test patterns available from evaluation procedures (e.g. Common Criteria) in order to discover potential weakness of any security products (as part of a vulnerability analysis) are candidates for security indicator test pattern. One approach is to follow/invest in identification of relation: GS ISI 001-1 [i.8]/SFRs [i.5] and SFRs [i.5]/Test pattern.

8.3 Generic vulnerability test patterns

8.3.0 Introduction on vulnerability test patterns

Eight vulnerability test patterns are proposed to be used for ISI stimulation: e.g. testing of countermeasures and detection means: sources e.g. DIAMONDS, EICAR:

- Verify audited event's presence
- Verify audited event's content
- Verify default-authentication credentials to be disabled on production system
- Verify presence/efficiency of prevention mechanism against brute force authentication attempts (active, passive)
- Verify presence/efficiency of encryption of communication channel between authenticating parties (active, passive)
- Usage of Unusual Behaviour Sequences
- Detection of Vulnerability to Injection Attacks
- Detection of Vulnerability to Data Structure Attacks

8.3.1 T1 - Test Pattern: Verify audited event's presence

Pattern name	Verify audited event's presence
Context	Test Pattern Kind: Behavioural Testing Approach(es): Detection
Problem/Goal	This pattern addresses how to check that a system logs a particular type of security-relevant event for auditing purpose
Solution	Test procedure template 1. Activate the system's logging functionality 2. Clear all existing log entries (if possible in the test environment) 3. Record current system time $\mathbf{t_s}$ 4. Stimulate the system to generate **the expected event type** 5. Check that the system's log contains entries for the expected event / Taking into account only logs displaying timestamps $\mathbf{t_l}$ satisfying following condition: $\mathbf{t_l > t_s}$
Known uses	Common Criteria SFRs [i.5]: FAU_GEN.1, FAU_GEN.2
Discussion	This pattern assumes that the test framework provides means for tracing and evaluating the logs produced by the SUT. Evaluation may be performed online (i.e. quasi simultaneously, while the system is still running) or offline, i.e. after the system has completed its operation. An interesting issue to consider is how to apply this pattern in situations whereby it may be impossible or too costly to clear the logs repository or to restart the running system.
Related patterns (optional)	• Sandwich test architecture pattern • Proxy test architecture pattern [i.14] • Verify audited event's content
References	[i.5], FAU_GEN.1, FAU_GEN.2

8.3.2 T2 - Test Pattern: Verify audited event's content

Pattern name	Verify audited events' content
Context	Test Pattern Kind: Behavioural Testing Approach(es): Detection
Problem/Goal	This pattern addresses how to check that a system logs a particular type of security-relevant event for auditing purpose
Solution	Test procedure template: 1. Activate the system's logging functionality 2. Clear all existing log entries / Record current system time t_s 3. Stimulate the system to generate the expected event type 4. Check that the system's log contains entries for the expected event / Taking into account only logs displaying timestamps t_l with $t_l > t_s$ 5. Store log entries containing the expected event type 6. Open the log entries and verify that their content meets the specified requirements
Known uses	Common Criteria SFRs [i.5]: FAU_GEN.1, FAU_GEN.2
Discussion	This pattern assumes that the test framework provides means for tracing and evaluating the logs produced by the SUT. Evaluation may be performed online (i.e. quasi simultaneously, while the system is still running) or offline, i.e. after the system has completed its operation. An interesting issue to be considered is how to apply this pattern in situations whereby it may be impossible or too costly to clear the logs repository or to restart the running system.
Related patterns (optional)	• Sandwich test architecture pattern • Proxy test architecture pattern • Extends test pattern *Verify audited event's presence* (Cf. Clause 8.3.1) by adding verification of the audited event's content.
References	CWE 311 [i.6]

8.3.3 T3 - Test Pattern: Verify default-authentication credentials to be disabled on production system

Pattern name	Verify default-authentication credentials to be disabled on production system
Context	Test pattern kind: Behaviour Testing Approach(es): Prevention
Problem/Goal	Enabled default authentication mechanisms, sometimes resulting from hard-coded credentials in source code are listed among MITRE's 2011 top 25 most dangerous software from the well-known CWE. For many software products, providing such a set of default authentication credentials is unavoidable, for example in a situation whereby some initial settings require an account on the system after it is installed. Although those default credentials are supposed to be modified before the system is actually deployed and made available to the outside world, several cases have been reported in which this was omitted, thus allowing attackers to bypass the authentication procedure and obtaining access to potentially sensitive data. This is particularly relevant for systems based on open-source software, given that the parameters for those default credentials are known to a large group of potential attackers. Therefore, providing test cases for detecting this kind of errors is very important for any software-based system with some authenticated interface to the outside world.
Solution	Test procedure template: Depending on whether a black-box or a white-box testing approach is applicable, different test procedures may be appropriate. Black-box testing procedure template: 1. Create (or reuse) a dictionary of default credentials usually available in open source software (e.g. login: admin, password: password; login: root; password: pass; etc.) 2. Try to authenticate using each time a new combination of credentials from the dictionary of step 1 3. If any of the authentication attempts is successful set FAIL verdict. Otherwise set PASS. White-box testing procedure template: 1. Create (or reuse) a dictionary of default credentials usually available in open source software (e.g. login: admin, password: password; login: root; password: pass; etc.) 2. Search the source code for any character string containing an element from the dictionary of step 1. Also include configuration files in the search. 3. If matching character strings are found, check that the source code implements a mechanism for enforcing the modification of authentication credentials.
Known uses	Amongst the DIAMONDS project case studies, the automotive case study has identified some elements of vulnerability derived from the weakness addressed by this test pattern: Several Bluetooth devices use "0000" as default PIN to access control. Therefore a test case verifying that the default PIN code has been replaced by a more user-specific one makes perfect sense in that context.
Discussion	If a black-box testing approach is chosen to apply this pattern, then it should be ensured that if present, a mechanism to block repetitive authentication attempts is deactivated, to avoid the SUT interpreting step 2 of the test procedure as a brute force hacking attempt, potentially leading to a cascade of other unwanted incidents unrelated with the actual test case.
Related patterns (optional)	• Mutually exclusive relation to pattern *Verify presence/efficiency of prevention mechanism against brute force authentication attempts* (Clause 8.3.4)
References	CWE 798 [i.6], OWASP-AT-003 [i.15]

8.3.4 T4 - Test Pattern: Verify presence/efficiency of prevention mechanism against brute force authentication attempts (active, passive)

Pattern name	Verify presence/efficiency of prevention mechanism against brute force authentication attempts
Context	Test pattern kind: Behaviour Testing Approach(es): Prevention, Detection
Problem/Goal	Password brute-forcing is a well-known attack pattern on computing systems providing a password-based authentication scheme (CAPEC 49 [i.7]).
Solution	Test procedure template: The mechanism for preventing may be passive, active or a combination of both. An example of passive mechanisms consist in adding elements on the authentication interface that cannot be interpreted automatically by a machine, but require human intervention. This is widely used in authentication forms on web-based interfaces in the form of so-called captchas, i.e. graphical images created dynamically, but designed in a way that makes them difficult to be read automatically by a computer program. The authenticating client is required to complete his/her credentials with the information encoded in the picture to ensure that a human being is well submitting the information. On the other hand, active mechanisms will initiate a series of steps to impede that the number of failed authentication attempts from the same source does not exceed a predefined threshold, beyond which appropriate steps are undertaken as counter-measures. The following test procedure template applies for an active prevention mechanism against password brute-forcing: Assuming that the maximal number of failed authentication attempts that triggers the defence mechanism is F_{max}, and that T_{max} is the maximal delay beyond which the defence mechanism is expected to come into play, proceed as follows 1. Use invalid credentials to authenticate on the system for F_{max} number of times or repetitively for a duration of T_{max} 2. Check that the SUT indicates that the used credentials are invalid and provides the user alternatives for the case he/she lost his/her credential details. 3. Optional: Check that failed authentication attempts are logged by the SUT and that the log entries contain as much information on the authentication source as possibly available. 4. Use invalid credentials once more to authenticate on the system 5. Check that the system reacts in a way that impedes a new authentication attempt unless certain steps are undertaken by the authenticating party (i.e. the test client). Possible reactions include: - (Temporarily) Blocking future authentication attempts from the same client. This assumes the authentication provider is able to clearly identify the source for the authentication request (e.g. using a combination of IP-Address, Host name, Operating System, MAC-Address, MSISDN, etc.) - Introducing additional hurdles to make successive authentication attempts from the same source more difficult, both technically and from a time and resource perspective.
Known uses	This security test pattern is widely used in all domains in which password-based authentication is applied (e.g. web-based applications and services, banking) Common Criteria SFRs: FIA_AFL.1 (Authentication Failures) [i.5].
Discussion	
Related patterns (optional)	• This pattern is applicable in cases whereby the *Authenticator* security pattern is used to ensure that entities accessing of a system are known as legitimate users thereof. • If the system logs all security-relevant incidents that occur at its external boundaries, as highly recommended by good practices in information systems security, then this pattern can be combined with the *Verify audited event's presence* pattern and the *Verify audited event's content* described in Clause 8.3.1 and Clause 8.3.2 respectively
References	CWE307 [i.6]

8.3.5 T5 - Test Pattern: Verify presence/efficiency of encryption of communication channel between authenticating parties (active, passive)

Pattern name	Verify presence/efficiency of encryption of communication channel between authenticating parties
Context	Test pattern kind: Behaviour Testing Approach(es): Prevention
Problem/Goal	Man-in-the-middle attacks are known to be among the most severe attacks an information system might face with regard to its security. One of the mitigation approaches consists in using encryption mechanisms (e.g. SSL) to protect the data exchange between authenticating parties from eavesdropping attempts with some of the numerous software tools freely available on the market and as open source.
Solution	Test procedure template: The steps to undertake for the test procedure are as follows: 1. Trigger the authentication client to start the authentication process using a well-known set of credentials. 2. Check that the monitoring test component has captured the packets exchanged between both authenticating parties. 3. Check that the captured packets do not contain any information as plain-text that could easily be read and understood by an attacker without a significant computation effort.
Known uses	FTP_ITC.1 (Trusted channel) [i.5]
Discussion	This test procedure is only applicable with a black-box testing approach and requires a testing architecture whereby an entity is positioned between both authenticating parties, with the ability to capture data traffic in both directions between them. This kind of architecture is based on the *monitoring test component* architectural pattern described in a previous FOKUS work on test patterns.
Related patterns (optional)	• Monitoring test component architectural pattern • CAPEC 94 [i.7]
References	

8.3.6 T6 - Test Pattern: Usage of Unusual Behaviour Sequences

Pattern name	Usage of Unusual Behaviour Sequences
Context	Test pattern kind: Behaviour Testing Approach(es): Prevention
Problem/Goal	Security of information systems is ensured in many cases by a strict and clear definition of what constitutes valid behaviour sequences from the security perspective on those systems. For example, in many systems access to secured data is pre-conditioned by a sequence consisting of identification, then authentication and finally access. However, based on vulnerabilities in the implementation of software systems (e.g. in the case of a product requiring authentication, but providing an alternate path that does not require authentication - CWE 288) [i.6], some attacks (e.g. Authentication bypass, CAPEC 115 [i.7]) may be possible by subjecting the system to a behaviour sequence that is different from what would be normally expected. In certain cases, the system may be so confused by the unusual sequence of events that it would crash. Thus potentially making it vulnerable to code injection attacks. Therefore uncovering such vulnerabilities is essential for any system exposed to security threats. This pattern describes how this could be achieved through automated testing.

Solution	Test procedure template: 1. Use a specification of the system to clearly identify the normal behaviour sequence it expects in interacting with an external party. If possible, model this behaviour sequence using a language such as UML, which provides different means for expressing sequenced behaviour, e.g. sequence diagrams or activity diagrams. 2. Run the normal behaviour sequence (from step 1) on the system and check that it meets its basic requirements. 3. From the sequence of step 1, derive a series of new sequences whereby the ordering of events would each time differ from the initial one. 4. Subject the system to each of the new behaviour sequences and for each of those: - Check that the system does not show exceptional behaviour (no live-/deadlock, no crashing, etc.) - Check that no invalid behaviour sequence is successfully executed on the system (e.g. access to secure data without authentication) - Check that the system records any execution of an invalid events sequence (optional)
Known uses	Model-based Behaviour fuzzing of sequence diagrams is an application of this pattern
Discussion	
Related patterns (optional)	
References	CWE 288 [i.6], CAPEC 115 [i.7])

8.3.7 T7 - Test Pattern: Detection of Vulnerability to Injection Attacks

Pattern name	Detection of Vulnerability to Injection Attacks
Context	Test pattern kind: Data Testing Approach(es): Prevention
Problem/Goal	Injection attacks (CAPEC 152 [i.7]) represent one of the most frequent security threat scenarios on information systems. They basically consist in an attacker being able to control or disrupt the behaviour of a target through crafted input data submitted using an interface functioning to process data input. To achieve that purpose, the attacker adds elements to the input that are interpreted by the system, causing it to perform unintended and potentially security threatening steps or to enter an unstable state. Although it could never be exhaustive, testing information systems resilience to injection attacks is essential to increase their security confidence level. This pattern addresses methods for achieving that goal.
Solution	Test procedure template: [i.7] 1. Identify all interfaces of the system under test used to get input with the external world, including the kind of data potentially exchanged through those interfaces. 2. For each of the identified interfaces create an input element that includes code snippets likely to be interpreted by the SUT. For example, if the SUT is web-based, programming languages and other languages frequently used in that domain (JavaScript, JAVA™…) will be used. Similarly, if the SUT involves interaction with a database, languages such as SQL may be used. The additional code snippets should be written in such a way that their interpretation by the SUT would trigger events that could easily be observed (automatically) by the test system. Example of such events include: - Visual events: e.g. a pop-up window on the screen - Recorded events: e.g. an entry in a logging file or similar - Call-back events: e.g. an operation call on an interface provided by the test system, including some details as parameters 3. Use each of the input elements created at step 2 as input on the appropriate SUT interface, and for each of those: - Check that none of the observable events associated to an interpretation of the injected code is triggered
Known uses	
Discussion	The level of test automation for this pattern will mainly depend on the mechanism for submitting input to the SUT and for evaluating potential events triggered by an interpretation of the added probe code.
Related patterns (optional)	• CAPEC 152 [i.7]
References	

8.3.8 T8 - Test Pattern: Detection of Vulnerability to Data Structure Attacks

Pattern name	Detection of vulnerability of data structure attacks
Context	Test pattern kind: Data Testing Approach(es): Prevention
Problem/Goal	Data structure attacks (CAPEC 255 [i.7]) consist in an attacker manipulating and exploiting characteristics of system data structures to violate the intended usage and protections of these structures and trigger the system to reach some instable state or expose further vulnerabilities that could be exploited to cause more harm. Detecting vulnerability to data structure attacks is among the key goals of security testing. The pattern provides a solution to that problem.
Solution	Test procedure template: 1. Identify all interfaces of the system under test used to get input with the external world, including the kind of data potentially exchanged through those interfaces. 2. For each of the identified interfaces create an input element including invalid values, i.e. values not meeting the requirements associated to their type and thus potentially unexpected by the SUT 3. Use each of the input elements created at step 2 as input on the appropriate SUT interface, and for each of those - Check that the SUT does not enter an unstable state at any time during the test case (no live-/deadlock, no crash, no exception, etc.)
Known uses	Data Fuzzing
Discussion	
Related patterns (optional)	• CAPEC 255 [i.7]
References	

8.4 Vulnerability test patterns based on MITRE

8.4.0 Introduction on vulnerability test patterns based on MITRE

The following test patterns are based on:

i) security issues raised from the description, common consequences, demonstrative examples, and observed CAPEC [i.7], CVE or CWE examples; and

ii) the definition of a list of keywords extracted from the Certification Commission for Health Information Technology (CCHIT) Ambulatory Criteria in order to help pointing a tester towards the correct security test pattern, [i.13]:

- Attacking a Session Management
- Attack of the authentication mechanism
- Testing the safe storage of authentication credentials
- Open Redirect
- Uploading a malicious file
- Searching for documented passwords
- Impersonating an external server
- Accessing resources without required credentials
- Ensuring confidentiality of sensitive information

8.4.1 T9 - Attacking a Session Management

Pattern name	Session Management Attack
Context	Testing Approach(es): behavioural and test data
Problem/Goal	This pattern addresses how to check that the system returns an authorization error when the session information is faked or forged, and that no sensitive information is returned after requests. Relevant for managing/controlling access the system.
Solution	Test Procedure Template: 1. Set up a proxy to monitor all HTTP or TCP traffic flowing to or from the server. 2. Authenticate to the system as a registered user. 3. Access one other page or screen (besides the home page or welcome screen) that requires authorization. 4. Log out. 5. Examine a captured HTTP request or TCP packet that is related to the access of the page other than the homepage. Identify headers or fields within the request or packet that may identify session identification information. 6. Modify a field identified in the earlier step (either by incrementing/decrementing them, removing them, replacing them with a different value entirely) and send this packet or request again. 7. Repeat the previous step for up to five fields identified in the packet or header. 8. Examine the cookies or local connection information (for systems that are not browser-based). Identify headers or fields within the cookie or local connection information that may identify session identification information. 9. Modify a field identified in the earlier step (either by incrementing/decrementing, removing, replacing with a different value entirely) and attempt to access the page or screen again without logging in. 10. Repeat the previous step for several other fields identified in the local connection information or cookies.
Known uses	Common Criteria SFRs [i.5]: FMT_MOF.1, FMT_MSA
Discussion	Since field modifications and resource access have to be performed, the evaluation of this pattern should be performed online. A difficulty would be to manage encryption on the platform as well as identification of relevant fields.
Related patterns	- Testing the safe transmission of authentication credentials - Modify Header Data - Modify Cookies or other Stored Information
References	CWE-311 and CWE-807 [i.6]

8.4.2 T10 - Attack of the authentication mechanism

Pattern name	Attacking Authentication Mechanism
Context	Testing Approach(es): detection, test data
Problem/Goal	This pattern addresses how to check that the system handles high number of authentication attempts with incorrect passwords. Relevant for authenticating multiple users through several simultaneous connections (performance).
Solution	Test Procedure Template 1. Write a script that captures and replays the sequence of HTTP or TCP signals for authenticating to the server. 2. Use this script to launch ten authentication requests with ten separate passwords from a list of frequently used passwords. 3. If the system attempts to block any of these incorrect authentication requests, check that there are no manipulatable fields in the headers or parameters involved in these requests that indicate the high number of the authentication requests. 4. Examine the request and response sequences for each of those HTTP or TCP signals and identify fields that may contain session identification information. 5. Run the script for 1000 connections simultaneously.
Known uses	Common Criteria SFRs [i.5]: FIA_AFL.1 and FIA_UAU.1
Discussion	The evaluation should be performed online. Difficulties: write a script capturing and replaying HTTP/TCP messages as well as searching for manipulatable fields. Some knowledge on the system under test is necessary.
Related patterns	- Test for Common Usernames and Passwords - Attacking the Authentication Nonce - Logging in more than X time - Obtain a Plethora of Connections
References	CWE-307, CWE-798, CWE-770 and CWE-327 [i.6]

8.4.3 T11 - Testing the safe storage of authentication credentials

Pattern name	Testing the safe storage of authentication credentials
Context	Testing Approach(es): detection
Problem/Goal	This pattern addresses how to check that the system store in a safe way the user authentication information. Relevant for user authentication management.
Solution	Test Procedure Template 1. Set up a connection to monitor all HTTP or TCP traffic flowing to the server or from the server. 2. Authenticate to the system as a registered user. 3. If the system is web-based, examine all cookies related to the system under test (e.g. by looking up its domain name). 4. Log out. 5. Access the system's database directly through a database management tool. 6. Find and view the table containing user authentication information (typically named similar to "users" or "user data").
Known uses	Common Criteria SFRs [i.5]: FIA_UAU.1, FIA_UID.1, FIA_UID.2
Discussion	The evaluation can be performed online or offline if the testing architecture is well defined. The information will be analyzed through the cookies and the user data. An efficient database management tool should be used to check the user authentication information.
Related patterns	
References	CWE-311 [i.6]

8.4.4 T12 - Open Redirect

Pattern name	Redirect header manipulation
Context	Testing Approach(es): design
Problem/Goal	This pattern addresses how to check that the system handles correctly the users' redirection after authentication. Relevant for URL parameters rejection.
Solution	Test Procedure Template 1. Set up to record HTTP traffic. 2. Authenticate as a registered user. 3. Browse to some pages other than the authentication page or homepage. 4. Observe the parameters sent to the web application in the URL. 5. Record any parameters that seem to indicate that the system is controlling where the user is to be redirected to after authentication. 6. Log out. 7. Manipulate the parameters recorded above to point to a dangerous or untrusted URL. 8. Log back in.
Known uses	CCHIT Criteria: AM 09.06 Common Criteria SFR [i.5]: FTP_ITI.1
Discussion	The evaluation can be performed offline after 'randomly' manipulating and monitoring the system. Some parameters have to be carefully defined before their modifications.
Related patterns	
References	CWE-601 [i.6]

8.4.5 T13 - Uploading a malicious file

Pattern name	Malicious file upload
Context	Testing Approach(es): test data
Problem/Goal	This pattern addresses how to check that the system should reject the file upon selection or should not allow it to be stored. Relevant for controlling stored or uploaded files.
Solution	Test Procedure Template 1. Authenticate as a registered user. 2. Open the user interface for action object. 3. Select and upload a malicious file in place of object. 4. View or download the malicious file.
Known uses	Common Criteria SFRs [i.5]: FDP_SDI.1, FDP_SDI.2 and FDP_ITC.1
Discussion	The evaluation can be performed offline after uploading a malicious file. The system should provide the ability to save scanned documents as images.
Related patterns	- Malicious file
References	CWE-434 [i.6]

8.4.6 T14 - Searching for documented passwords

Pattern name	Search for documented passwords
Context	Testing Approach(es): detection, test data
Problem/Goal	This pattern addresses how to check that the system should not list any default passwords or usernames that are hard-coded into the product.
Solution	Test Procedure Template 1. Search the system's documentation. 2. Look at the HTML or any marked-up text that is included with the system by default.
Known uses	Common Criteria SFRs [i.5]: FPT_ITI.1 and FPT_ITC.1
Discussion	The evaluation is performed offline. A difficulty is the identification of the elements (users' information or password).
Related patterns	
References	CWE-798 [i.6]

8.4.7 T15 - Impersonating an external server

Pattern name	Impersonating trusted external resources
Context	Testing Approach(es): design, data
Problem/Goal	This pattern addresses how to check that the system refuses (or behave as such) to connect an impersonated server. Indeed, by DNS spoofing or DNS entry modifications, the authentic external server may be replaced. Relevant for checking trusted path/channels.
Solution	Test Procedure Template 1. Set up a connection to monitor all HTTP or TCP traffic flowing to the server or from the server. 2. Authenticate as a registered user. 3. Open the user interface to action an object. 4. Identify any request that was sent to an external server and record it. 5. Impersonate the external server, either by changing the settings of the system to point to that server or by DNS spoofing the external server and replacing that DNS entry with the one of the impersonated server. 6. Construct a response from the impersonated server that performs the same functionality as the authentic external server. 7. Open the user interface to action an object again. 8. Log out.
Known uses	Common Criteria SFRs [i.5]: FTP_ITC.1 and FTP_TRP.1
Discussion	The evaluation is performed online. A difficulty could be the response to be built and sent by the tester.
Related patterns	- DNS Spoofing an Update Site - Pointing to an Untrusted Update Site - Spoofing Functionality Provided in Untrusted Sphere
References	CWE-494 and CWE-829 [i.6]

8.4.8 T16 - Accessing resources without required credentials

Pattern name	Exposing functionality requiring authorization
Context	Testing Approach(es): design
Problem/Goal	This pattern addresses how to check that the system disallows a user to action an object if she has not the proper credentials.
Solution	Test Procedure Template 1. If access to action an object requires authentication, authenticate as a registered user. 2. Open the user interface, either inside or outside of the main application, for actioning the object. 3. Record the series of mouse clicks, GUI interactions, or URL sequences required to get to this screen. 4. Log out and/or exit this screen. 5. Attempt to repeat the series of steps recorded above.
Known uses	Common Criteria SFRs [i.5]: FDP_ACC.1, FDP_ACC.2 and FDP_ACF.1
Discussion	The evaluation is performed online while an analysis of the performed actions can be made offline. Some actions could be difficult to automate (forms in which to enter data, specific values to provide through a database process). It will depend on the design of the user interface.
Related patterns	Exposing Critical Functionality Force Exposure of Function Requiring Authorization
References	CWE-306 and CWE-862 [i.6]

8.4.9 T17 - Ensuring confidentiality of sensitive information

Pattern name	Sensitive information confidentiality
Context	Testing Approach(es): architectural
Problem/Goal	This pattern addresses how to check that the system use a known safe encryption protocol. Relevant to test if any sensitive or personal information contained within an object is only accessible to the user who actioned it.
Solution	Test Procedure Template 1. Authenticate as a registered user. 2. Open the user interface for actioning an object. 3. If necessary, open, view, or otherwise access the actioned object. 4. Log out.
Known uses	Common Criteria SFRs [i.5]: FCS_COP.1
Discussion	The evaluation is performed either online or offline. Some actions on the objects could provide different behaviours that could eventually relate on other test patterns. Finally some expected results could be that: i) the connection to the server was made using a known safe encryption protocol (e.g. HTTP over SSL, or an encrypted TCP connection); and ii) the manipulated object is encrypted with a safe encryption protocol, password-protected, or both.
Related patterns	- Testing the safe transmission and storage of sensitive personal information - Testing the safe transmission of sensitive data to an outside source - Force the Export of Sensitive Information
References	CWE-311, CWE-212 [i.6]

8.5 Mapping of vulnerability test patterns with ETSI GS ISI 001-1 indicators

8.5.0 Introduction

Tables 1 to 3 summarize the mapping between the vulnerability test pattern examples and the family of indicators extracted from ETSI GS ISI 001-1 [i.8]. Links with the security incidents (Ixx) covers cases where the vulnerability test campaign has to be detected as attacks by the detection system. Links with the security vulnerabilities (Vxx) covers cases where the vulnerability test campaign tries to find vulnerabilities that detection system did not detect before.

8.5.1 Security Incidents (Ixx)

Table 1: Mapping between test cases and Security Incidents (Ixx)

CLASS	FAMILY		COMPONENT AND IDENTIFIER		TESTCASES
IEX intrusions and external attacks	FGY	Website forgery	1 2	**Forged domain or brand names** **Forged websites**	T15
	PHI	Phishing	1 2	**Targeting customers' workstations** **Targeting organization's users**	T15
	INT	Intrusion	1 2	**Attempt on externally accessible servers** **Success on externally accessible servers**	T4, T6, T7, T8, T9, T10
	DFC	Website defacement	1	**Obvious and visible website defacements**	T7, T8
	MIS	Misappropriation of resources	1	**Servers resources misappropriation** (by external attackers)	T6, T16
	SPM	Spam	1	**Messages targeting org. users**	
	DOS	Denial of Service	1	**DoS and DDoS attacks on websites**	T6
	MLW	Malware	1 2 3 4	**Attempts to install malware on workstations** **Attempts to install malware on servers** **Installations on workstations** **Installations on internal servers**	All components: T6, T13, T16
	PHY	Physical intrusion or action	1	**Human intrusion into organizations perimeter**	

CLASS	FAMILY		COMPONENT AND IDENTIFIER		TESTCASES
IMF malfunctions	BRE	Accidental breakdowns or malfunctions	1 2 3 4	**Workstations breakdowns or malfunctions** **Servers breakdowns or malfunctions** **Mainframes breakdowns or malfunctions** **Networks breakdowns or malfunctions**	T6
	LOM	Loss or theft of mobile devices	1	**Mobile devices belonging to org.**	
	TRF	Trace malfunction	1 2 3	**Downtime or malfunction of trace production** **Absence of possible tracking of involved person** **Downtime/malfunction of trace production for recordings with evidential value**	All components: T1, T2

IDB Internal deviant behaviour	UID	Identity usurpation	1	User impersonation	T9, T10
	RGH	Rights (or privileges) usurpation or abuse	1 2 3 4 5 6 7	Privilege escalation by exploitation of software or config vul. Privilege escalation by social engineering Use of admin rights illicitly granted by admin Use of time-limited rights after period Abuse of privileges by admin Abuse of privileges by operator or user Illicit use of rights not removed (after departure or position change)	T16 T6, T11
	IDB	Other incidents (reg. unauthorized access)	1	Unauthorized access to servers through remote access points	T6
	MIS	Misappropriation of resources	1	Server resources misappropriation by an internal source	T6, T16
	IAC	Illicit access to Internet	1	Access to hacking website (from internal workstation)	T6
	LOG	Deactivating of logs recording	1	Deactivating of logs recording by an admin	T1, T2

IWH whole incident class	VNP	Non-patched or poorly patched vul. exploitation	1 2 3	Exploitation of sw vul. w/o available patch Exploitation of non-patched sw vul. Exploitation of poorly-patched sw vul.	
	VCN	Conf. vul. exploitation	1	Exploitation of config flaw	T3, T4, T14
	UKN	Unknown incidents	1	Not categorized sec incidents	
	UNA	Incidents on not addressed assets	1	Sec. inc. on non-inventoried/not-managed assets	

8.5.2 Indicators with vulnerabilities (Vxx)

Table 2: Mapping between test cases and vulnerabilities (Vxx)

CLASS	FAMILY		COMPONENT AND IDENTIFIER		TEST CASES
VBH Behaviour vulnerabilities	PRC	Dangerous protocols used	1 2 3 4 5 6	Server accessed by an admin with unsecure protocols P2P client in a workstation VoIP client in a workstation Outbound connection dangerously set up Not compliant lap top computer used to establish a connection Other unsecure protocols used	T5 T5 T6
	IAC	Internet illicit access	1 2	Outbound controls bypassed Anonymization site used	T6
	FTR	File illicit transfer with outside	1 2 3	File recklessly downloaded Personal public instant messaging account used (for business file exchanges) Personal public messaging account used (for business file exchanges)	

CLASS	FAMILY		COMPONENT AND IDENTIFIER		TEST CASES
	WTI	Workstation used w/o relevant usual security	1 2 3 4 5 6	Workstation with a disabled or not updated AV and/or FW Workstations accessed in admin mode Personal storage devices used Personal devices used w/o compartmentalization (BYOD) Not ciphered sensitive files exported Personal software used	T17
	PSW	Passwords illicitly handled or managed	1 2 3	**Weak passwords used** **Passwords not changed** Admin passwords not changed	All components: T3, T10, T14
	RGH	Access rights illicitly granted	1	No compliant user rights granted by admin	T16
	HUW	Human weakness	1 2	Exploited by spear phishing message (links/attachments) By exchanges secrets (phone/f2f)	
VSW Software vul.	WSR	Webserver sw. vul.	1	**Web applications sw vul.**	T4, T5, T7, T8, T9, T10, T11, T12, T17
	OSW	OS sw. vul	1	**OS sw vul. regarding servers**	T4, T5, T11, T17
	WBR	Webbrowser sw. vul.	1	**Webbrowser sw. vul.**	T5, T8, T11, T17
VCF Configuration vul.	DIS	Dangerous or illicit services	1	Dangerous or illicit services on externally accessible servers	
	TRF	Log production shortcomings	1	Insufficient size of the space allocated for logs	T1, T2
	FWR	Weak FW config.	1	Weak FW filtering rules	
	ARN	Autorun feature enabled	1	Autorun feature enabled on workstations	
	UAC	User accounts wrongly configured	1 2 3 4 5	Access rights configuration not compliant with security policy Not compliant access rights on logs Generic and shared admin account Accounts w/o owners Inactive accounts	T3, T14, T16
VTC General sec. technical vul.	IDS	IDS/IPS malfunction	1	Full unavailability of IDS/IPS	
	WFI	Illicit Wi-Fi access points	1	Wi-Fi devices installed on the network w/o any official authorization	
	MOF	Poor monitoring	1	Absence or poor quality of monitoring of some outgoing flows	T1, T2, T5
	RAP	Illicit remote access	1	Remote access points used to gain unauthorized access	
	NRG	Illicit network connections	1	Devices or servers connected to org. network w/o being reg./managed	
	PHY	Physical access control	1	Not operational phy. access control means	
VOR General sec. org. vul.	VNP	Not patched vul.	1 2	Excessive duration of windows of exposure Rate of not patched systems	
	VNR	Not reconfigured systems	1	Rate of not reconfigured systems	
	RCT	Reaction plans	1 2	Reactions plans launched w/o experience feedback Reaction plans unsuccessfully launched	
	PRT	Security in IT projects	1 2 3	Launch of new IT projects w/o information classification Launch of new specific IT projects w/o risk analysis Launch of new IT projects of a standard type w/o identification of vul. and threats	

8.5.3 Indicators as regards impact measurement (IMP)

Table 3: Mapping between test cases and impact measurement (IMP)

IMP	COS	Costs	1	Average cost to tackle a critical sec. incident	
	TIM	Average time of website downtime	1 2 3	Due to whole sec incidents Due to successful malicious attacks Due to malfunctions/unintentional sec. incidents	

Annex A (informative): Authors & contributors

The following people have contributed to this specification*:*

Rapporteurs:

Christophe Blad, Oppida

Axel Rennoch, Fraunhofer Fokus

Other contributors:

Herve Debar, Institut Telecom, Vice-Chairman of ISG ISI

Gerard Gaudin, G²C, Chairman of ISG ISI

Stéphane Lu, BNP Paribas

And in alphabetical order:

Eric Caprioli, Caprioli & Associés

Paolo De Lutiis, Telecom Italia

Jean-François Duchas, Bouygues Telecom

Christophe Delaure, Qualys Inc.

François Gratiolet, Qualys Inc.

Stéphane Lemée, Cassidian (an EADS company)

Federic Martinez, Alcatel-Lucent, Secretary of ISG ISI

Jean-Michel Perrin, Groupe La Poste

Laurent Treillard, CEIS

Annex B (informative): Bibliography

Club R2GS 4-page data sheet V3 (2012): "Presentation of the work in progress" (accessible on ETSI ISG ISI portal).

Club R2GS presentation V4 (March 2012): "The Club and its objectives" (accessible on ETSI ISG ISI portal).

Club R2GS reference framework V1.0 (December 2009): "A security event classification model at the heart of SIEM approaches" (accessible on ETSI ISG ISI portal).

History

Document history		
V1.1.1	November 2015	Publication