Netherlands Scheme for Certification in the Area of IT Security (NSCIB)

Nederlands Schema voor Certificatie op het gebied van IT-Beveiliging (NSCIB)

NSCIB Scheme Procedure #6

Evaluator Reporting during the Monitoring Phase

Rob Huisman

Approved (signature on file)
Technical Manager NSCIB

Procedure	6
Report title	Evaluator Reporting during the Monitoring Phase
Date of issue	31 January 2020
Version	2.0
Filename	NSP_06_Evaluator_Reporting_during_the_Monitoring_Phase_v2.0.docx

TÜVRheinland®
Precisely Right.

Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

Document history

Version	Date	Comment
1.0	June 1, 2012	1st release for EAL3
1.1	July 1, 2013	Draft with EAL4 extensions
1.2	May 20, 2014	2nd Draft after internal comments
1.3.1	September 23, 2016	3rd Draft with alternative ADV and ATE approaches.
1.3.2	October 20, 2016	Added some EAL5 elements. Comments on 1.3.1 incorporated
1.4	July 2019	Updates to consider addition of EAl5-7
1.5	August 23, 2019	Title changed. EAL5-7 elements added. Reformulated to being the standard method of reporting.
1.6	7 October 2019	Final draft for “fit for purpose” review, including industry partners.
1.7	16 October 2019	Reviewed within NSCIB scheme
2.0	31 January 2020	2nd release

References

[AIS 31/20]	Functionality classes and evaluation methodology for deterministic/physical random number generators, version 3, 15.05.2013, Bundesamt für Sicherheit in der Informationstechnik
[CC]	Common Criteria for IT Security Evaluation, Part 1,2 and 3, v3.1r5, April 2017
[CCRA]	Arrangement on the Recognition of CC Certificates in the field of IT Security, July 2014
[CEM]	Common Methodology for IT Security Evaluation, v3.1r5, April 2017
[CoDE]	Collection of Developer Evidence, v1.5, JIL, January 2012
[JIL_COMP]	Composite product evaluation for Smart Cards and similar devices, v1.5.1, JIL, May 2018
[SOGIS_MRA]	SOGIS MRA of IT Security Evaluation Certificates, v3.0, January 2010

Glossary of terms and abbreviations

This list does not contain terms already defined by the [CC] or [CEM].

NSCIB	Netherlands Scheme for Certification in the Area of IT Security (Nederlands Schema voor Certificatie op het gebied van IT-Beveiliging)
NSP	NSCIB Scheme Procedure
EWP	Evaluation Work Plan
EM1/2/3	First, Second, and Final Evaluation Meeting

Table of contents

1 Introduction

Previous procedures in the NSCIB were oriented towards:

1. “The developer shall provide developer documentation that completely and consistently complies with every aspect of every relevant work unit of the CEM, and does not contain any ambiguity whatsoever”
2. “The lab shall document for each work unit a complete rationale and a summary of the evidence so that the certifier can ascertain that each work unit has been completely and correctly performed and each piece of CEM guidance has completely and correctly been applied.”

This was a significant amount of work for the developer who had to learn in depth about the CC (or pay for a consultant to completely rewrite all documentation), for the lab, which had to write very extensive reports, and for the CB who then had to check it all. Often there is more than one feedback cycle (caused either by lab or CB), causing more work and delay.

In the international area, point 1 has been acknowledged, and as a reaction, the document “Collection of Evidence” [CoDE] was developed to support the evaluator extracting information from developer documentation and interviews with the developer rather than having the developer/consultant rewrite the documentation.

Earlier versions of this NSCIB Scheme Procedure (NSP) aimed at improving point 2 above and presented a standard method of reporting from evaluator to CB. The goal was for the evaluator to:

- continue to perform all activities mandated by [CC], [CEM] and NSCIB Scheme Procedure #1 and NSCIB Scheme Instructions;
- spend less time on reporting these results, by adopting an efficient *style* of reporting;
- spend more time on testing and especially penetration testing.

The approach is generally described in Chapter 2 with references to the Chapters 4-14 which describe the more detailed procedures.

This version of the NSP also addresses the improvement of point 1 above, to allow the developer, in very specific cases to provide much less documentation to the evaluator than would otherwise be the case. This is referred to as the “alternative method”.

In Appendix A provides an example mapping between the CC evaluator actions and where in the evaluator reports they are demonstrated where in the evaluator reports they are demonstrated.

2 Process overview

The general process consists of the following 8 steps:

1. Application
2. Kick-off meeting (optional) (Preparation Phase: detailed in NSP#1)

----

3. ST Evaluation
4. First Evaluation Meeting
5. Second Evaluation Meeting (Monitoring Phase: detailed in this scheme procedure)
6. Final Evaluation Meeting
7. Final Evaluation Reporting

----

8. Certification Phase (Certification Phase: detailed in NSP#1

Steps 1, 2 and 8 are described in NSP#1. Steps 3 to 7 are explained below.

For some evaluations, the First and Second Evaluation Meetings may be combined into one (1) meeting where all the required deliverables for EM1 and EM2 will be presented in one go. The combination of the First and Second Evaluation Meeting is the default when the Alternative ADV method is used because for this method a separation of the design-level and the implementation-level assessment is counter-intuitive.

Combining the First and Second Evaluation Meetings needs to be decided during the Application process because the reduction of the number of meetings reduces the certification costs. Factors which are relevant for the decision are for example the complexity of the product in combination with the required level of assurance. The amount of material and discussion for complex products may mean that the meeting has to stretch into a second day of discussion.

2.1 ST Evaluation

The developer provides a more-or-less final ST. The lab adds the accompanying evaluation report with the ASE evaluation results to the certifier. After one or more feedback loops based on official review reports as detailed in NSP#1 the ST is provisionally[1] accepted by the certifier (approval by the NLNCSA Monitoring certifier is only added at the time of final evaluator reporting).

2.2 General Requirements for the Evaluator Presentations

With the exception of the Security Target (ASE) evaluation results (as discussed above), the evaluator reporting within NSCIB evaluations is based on evaluation meetings and presentations. In the meeting deliverables the evaluator shows *how* all content and evaluator action elements for the processing of the assurance components that are relevant for the evaluation are met. This must be done within the presentation, and may additionally be supported by an annex or other evaluator analysis documents. The evaluator shall also provide a checklist of where evaluator action items are demonstrated in the meeting deliverables, to a level of content and presentation elements.

At EM1 the checklist for the entire assurance level shall be presented, populated as appropriate for EM1. This document is then further populated for subsequent evaluation meetings. This means that the checklist presented in EM3 will be completely populated and should contain only 'pass' verdicts.

When methodology applied by the evaluator describes explicit reporting to be provided, this explicit reporting needs to be automatically provided as part of the evaluation documentation. For example, both [AIS 31/20] and [JIL_COMP] explicitly describes workunits to be performed, and this needs to be reported on a workunit level, either in a separate document or alternatively in the evaluation meeting presentation. The reporting should be added into the appropriate meetings. For example, the AIS20/31 reporting, with the analysis and testplan (without the test results) needs to be delivered and presented for EM2 at the latest. The AIS20/31 test results need to be described in EM3.

[1] There is always room for later changes due to developer changes, new information coming to light or new insights at lab, developer or CB. Note that this term "provisionally approved" is used throughout this document.

2.3 First Evaluation Meeting

2.3.1 First Evaluation Meeting Procedure

The laboratory organizes a meeting with the certifier at a mutually agreed location. The developer is encouraged, but not required, to attend the meeting. Other parties are only allowed to attend if developer, lab and certifier agree.
Five working days before the meeting the laboratory will send all First Evaluation Meeting Deliverables (see 2.3.2) to the certifier.

In this meeting the First Evaluation Meeting Deliverables are presented by the evaluator, according to the following rules:

- Not all First Evaluation Meeting Deliverables actually need to be presented. As the certifier has had one week to study the deliverables, he/she may allow the evaluator to skip certain sections that the certifier deems to be self-explanatory.
- The certifier is allowed to question the evaluator on any or all of the items to ascertain that the evaluation was performed correctly and completely.
- If there are any missing items in the First Evaluation Meeting Deliverables, or items that are not clear, these will be corrected during the meeting, by amending the First Evaluation Meeting Deliverables where possible and annotating them where amending would take too much time.

The meeting can have four possible outcomes:

1. All First Evaluation Meeting Deliverables were either correct or successfully amended/annotated during the meeting. In this case all of these deliverables are provisionally approved.
2. One or more deliverables could not be successfully amended/annotated, but the certifier determines that this can be further handled by email. In this case, the other deliverables are provisionally approved, and after an email process, where the remaining deliverables are amended/annotated will also be provisionally approved.
3. One or more deliverables could not be successfully amended/annotated and cannot be handled by email, but the certifier determines that this can be rescheduled to the Second Evaluation Meeting. In this case, the other deliverables are provisionally approved, and the remaining deliverables are rescheduled.
4. One or more deliverables could not be successfully amended/annotated and the certifier determines that this cannot be handled by email or rescheduling. In this case, the entire First Evaluation Meeting is nullified, and must be repeated once the evaluator has remedied everything.

The laboratory will take notes of all decisions made and issues raised by the certifier where further action is required (i.e. deliverable annotated during meeting, further handling by email or renewed discussion of the issue at a rescheduled meeting). This list with issues and related actions will be emailed to the certifier for confirmation within 2 working days after the meeting.

No full meeting minutes or detailed review reports are required, but the certifier will endeavour to provide a written list of issues before, or at the meeting.

2.3.2 First Evaluation Meeting Deliverables

The First Evaluation Meeting Deliverables consist of the following:

- Checklist of all evaluator action items and content and presentation elements relevant for the claimed assurance level (populated to show where the evaluator actions and c&p elements relevant to EM1 are demonstrated).
- Updated ST and IR_ASE according to certifier comments;
- The ADV Presentation (see Chapter 4);
- The Implementation Representation Sampling Rationale (see Chapter 5);
- The ADV/AGD Reference Document (see Chapter 6) and all guidance documents that this document refers to;
- The Configuration Item Identification Presentation (see Chapter 7);
- The Consultancy/Evaluation Improvement Presentation (see Chapter 14);
- Any other observations that were found before this meeting and are deemed relevant.

2.4 Second Evaluation Meeting

2.4.1 Second Evaluation Meeting Procedure

The Second Evaluation Meeting Procedure is identical to the First Evaluation Meeting Procedure, with the exceptions that it concerns different deliverables (see 2.4.2).

2.4.2 Second Evaluation Meeting Deliverables

The Second Evaluation Meeting Deliverables consist of the following:

- Updated Checklist showing where the evaluator actions and c&p elements relevant to EM1 and EM2 are demonstrated;
- Any First Evaluation Meeting Deliverables that were rescheduled to this meeting;
- The Implementation Representation Presentation (see section 8);
- The ATE/AVA Test plan Presentation (see section 9);
- The ATE/AVA test descriptions (see section 10);
- The ALC Presentation, including ALC verification plan (see section 11);
- Any other observations that were found before this meeting and are deemed relevant.

2.5 Final Evaluation Meeting

2.5.1 Final Evaluation Meeting Procedure

This procedure is identical to the First and Second Evaluation Meeting Procedure, with the following exceptions:

- It concerns different deliverables (see 2.5.2)
- It cannot end with outcome #3 (see 2.3.1) as there is no meeting to reschedule to.

2.5.2 Final Evaluation Meeting Deliverables

The Final Evaluation Meeting Deliverables consist of the following:

- Completed Checklist showing where all evaluator actions items and content and presentation elements relevant for the claimed assurance level are demonstrated;
- Any Second Evaluation Meeting Deliverables that were rescheduled to this meeting;
- The final ST (and ST-Lite if applicable);
- The final guidance documentation for the TOE satisfying AGD_PRE and AGD_OPE.
- The ATE/AVA test results (see section 12);
- The ALC Results Presentation and draft STAR (if applicable) (see section 13);
- Draft ETR, draft ETRfc (if applicable);
- Any further observations that were found before this meeting and are deemed relevant.

2.6 Final Evaluation Reporting

The laboratory delivers its revised Evaluation Technical Report (ETR) and the revised (final) versions of the evaluation deliverables of the final meeting and, if this is found correct; the certifier formally approves the ETR and all provisionally approved items.

3 Notation

In the following chapters, the following notation is used:

Evaluator presentation actions (the actions an evaluator has to do) are always encased in a green box.

This reporting is not “complete” in the sense that it reports every CEM detail at the level of a work unit. However, in the NSCIB this, together with the checklist mapping where the evaluation action items and content and presentation elements are reported, is sufficient to meet the reporting requirements indicated in the green box. Note that this does *not* allow the evaluator <u>not to use</u> the CC or CEM: this is only intended for what needs to be reported. Any further recording of results is left to the lab and to the ISO-17025 standard.

Often these boxes are then followed by an example, to illustrate some important concept.

Finally, a short summary of the result is then given. This result is always encased in an orange box.

4 The ADV Presentation

The overall goal of the assurance class ADV is for the evaluator to understand the TOE to the level that he can understand how it implements security, and to assist the evaluator in determining his tests and penetration tests.
The role of the certifier is to ascertain that the evaluator understands the design (and has done all the work). To this end, while the presentation may contain useful examples from the developer evidence, the presentation should not just be comprised of copied material from the developer evidence. Rather it should reflect the evaluators' summary of that material with appropriate references.

The ADV presentation will present the following elements:

- o The TOE and the TSFI
- o Subsystems
- o Modules
- o Tracing SFRs to TSFI and Subsystems
- o Security Architecture
- o Other items based on applicable mandatory methodology (e.g. [JIL_COMP])

For the evaluation (and presentation) of ADV under this NSP#6, there exist two methods:

1. The regular ADV method,
2. The alternative ADV method.

The regular ADV method is based on evaluator analysis of a full set of developer evidence to meet each and every developer action item (down to the level of content and presentation elements).

The alternative method for ADV is an extension of the Collection of developer evidence process, using implementation representation as a basis. This approach can only be used in cases where the laboratory has a vast experience with the TOE type in question and is able to determine the full TSF security behaviour from the implementation representation. The regular ADV method is to be used in all other cases.

In order for a laboratory to use the alternative ADV method, three conditions must be met:

- The CB must give permission. Therefore, the use of this method must be documented in the EWP.
- Even if ADV_IMP.1 is claimed, the <u>entire</u> implementation representation must be made available to the evaluator and sufficiently annotated with informal text to enable the evaluator to trace all SFRs to the modules, as defined in the implementation representation.
- The alternative method for ATE must be used (see section 9.2).

The alternative method exploits the fact that the laboratory is so familiar with the TOE type that the laboratory can:

- Perform a vulnerability analysis directly on the implementation representation, without requiring detailed TDS-type developer evidence.
- Determine whether the SFRs are met by the implementation representation, without requiring detailed ADV_TDS-type developer evidence.
- Determine whether the constructs described in the developer ARC document are correctly implemented, without requiring detailed TDS-type developer evidence.

Under those three conditions, the whole of ADV_TDS is considered to be defined by the implementation representation, that is:

- Modules are sets of implementation representation (e.g. source code, VHDL), and the interfaces of those modules are the interfaces of that implementation representation Since the modules are defined by the implementation representation they automatically meet any semi-formal description requirements required for the evaluation assurance level.
- The evaluator uses his vast experience with the TOE type in question to identify all SFR-enforcing and SFR-supporting modules as part of the ADV_IMP work. The entire implementation representation must be described at a level as if it is SFR-enforcing. A summary of this identification is provided by the evaluator in the form of an overview of the TOE and how it implements the SFRs. While the full mapping needs to be completed in order to ensure the necessary modules are identified for ADV_TDS, there is no need to present the

full mapping of the SFRs to the modules. The presentation must provide an example of how this mapping is generated and, on demand, the evaluator must be able to show how a specific SFR is implemented by the modules.
Subsystems are sets of modules and the interfaces of those subsystems are the externally accessible interfaces of the modules. If the modules are sufficiently described, then also the subsystems are sufficiently described and additional subsystem level descriptions are not required.

If all SFRs can be traced to the implementation representation, and the implementation representation meets the ADV_IMP.1/ADV_IMP.2 requirements (as considered in section 8.2 or 8.3 as applicable for the assurance level), all ADV_TDS requirements are met and need not be checked separately or described further by the evaluator. The only evaluator activity required is the presentation of the method used by the evaluator to identify the modules from the sets of code. This description should be accompanied with examples of the identified modules and rationale of how they fit the method for identifying modules.

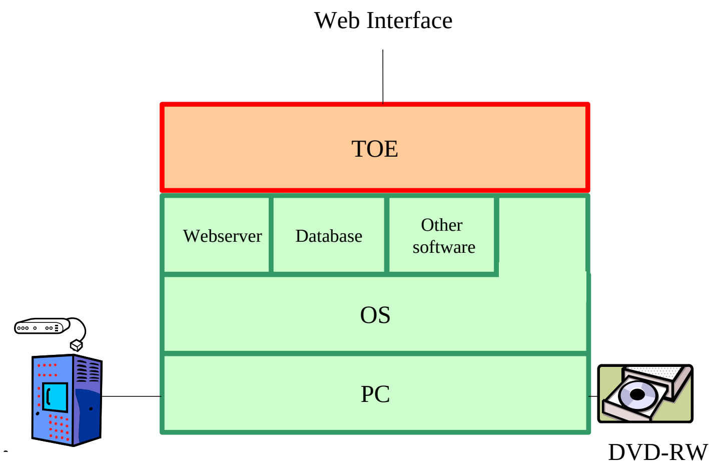
4.1 The TOE and the TSFI

This section applies to both the regular and the alternative ADV method.

Observe that the developer has to present for ADV_FSP.6 a formal model for the TSFI which has to be addressed in addition. Refer to Section 4.6 for further details. In this case the alternative ADV method cannot be applied.

1. The evaluator presents a model of the TOE in its environment:
   - where necessary, this model shall be supplemented with photos of the TOE or the actual TOE;
   - this model shall clearly show all interfaces of the TOE;
   - all interfaces shall be explained as TSFI or non-TSFI (also note NSI#7 for a detailed definition of TSFI);
   - the purpose and method of use of all TSFI shall be presented;
   - this model shall show all user roles that interact with each TSFI, and where useful, all other interfaces.
2. The evaluator explains how he determined completeness.

Example of a model:



The only TSFI is the Web Interface (defined in [FSP] section x.y). The interface with the DVD-RW, and other external boxes are not TSFI, as they are B1 interfaces. The interfaces to Webserver, Database, Other Software, OS, and PC are not TSFI, as they are B2 interfaces. See CC Part 3 Annex A.2.2.[2]

[2] See also TSFI definition in NSCIB Scheme Instruction #09: 'Clarification of the TSFI'.

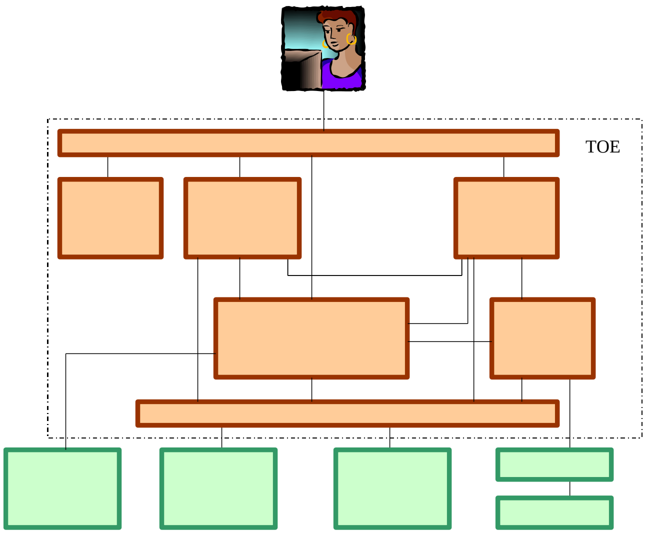
Result: The evaluator demonstrates that all interfaces and TSFI have been identified and described.

4.2 Subsystems

4.2.1 The regular ADV method for subsystems

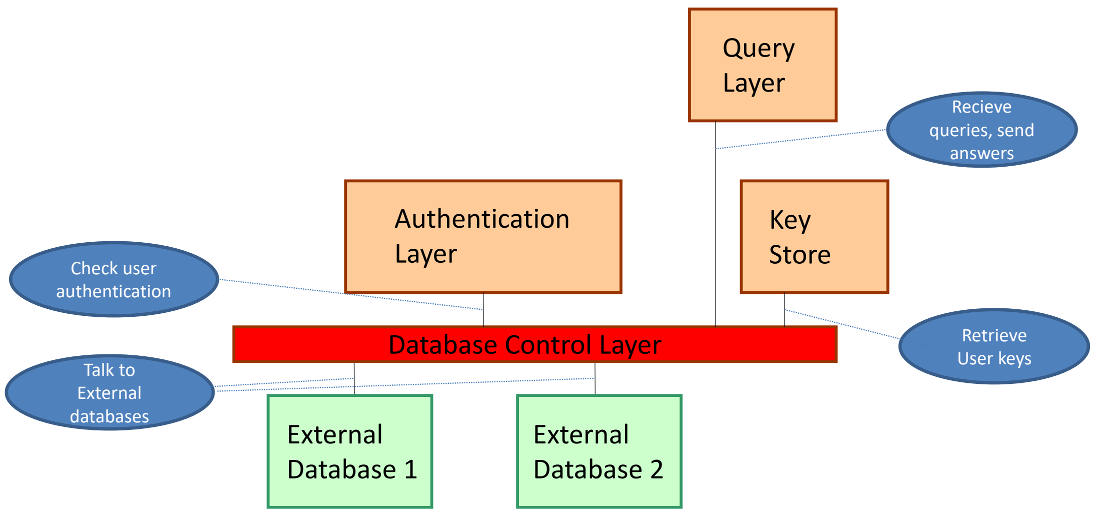
1. The evaluator presents a subsystem level model of the TOE (possibly with some parts of the environment):
    - this model shall be sensible and useful[3] [4];
    - this model shall show all TSFI, and where useful, all other interfaces;
    - this model shall clearly clarify whether subsystems are TOE, TSF or environment and whether they are SFR-enforcing, SFR-supporting or SFR non-interfering.
2. The evaluator explains the behaviour of each subsystem and its interaction with other subsystems. This explanation shall make use of examples from the developer evidence (e.g. diagrams).

Example of the subsystem level model:



Example of the subsystem behaviour and interaction (of the red subsystem)

[3] The current CC allows and some schemes advocate the use of “stupid” designs for EAL2, which have e.g. one TSFI per SFR and one subsystem per TSFI. These add nothing to understanding.
[4] It is highly recommended that the evaluator presents a model that is (closely related to the model) used by the developer. The model presentation shall include references to the relevant sections of the developer evidence.



Result: The evaluator demonstrates that he understands the TOE design and that it identifies and describes all subsystems

4.2.2 The alternative ADV method for subsystems

In the alternative ADV method for subsystems, all requirements for the subsystems are met by the implementation representation. As noted early in section 4, subsystems and their interfaces are sets of modules. Hence, if the modules are sufficiently described then by inference any subsystem from which they are derived are also sufficiently described. Therefore, no further evaluator actions to those specified in section 4.3.2 are required at this point.

Observe that the developer has to present for ADV_TDS.6 a formal model for the TSF subsystems which cannot be addressed under the alternative ADV method. Refer to Section 4.6 for further details.

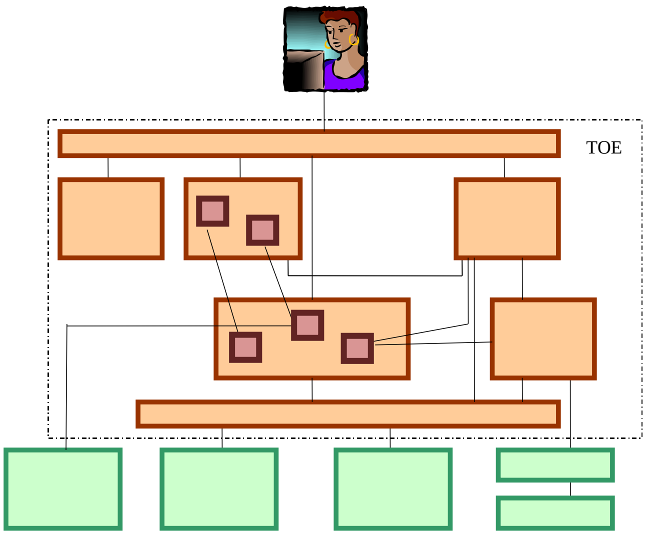
4.3 Modules

4.3.1 The regular ADV method for modules

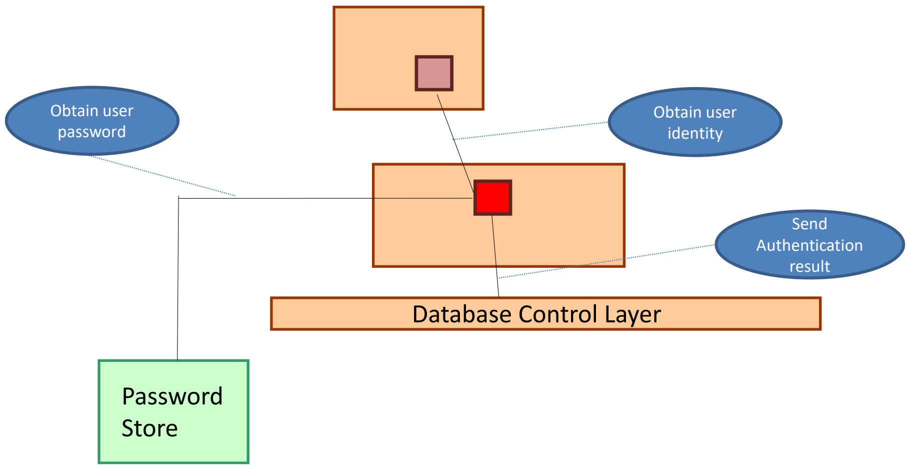
1. The evaluator presents a module level model of the TOE (possibly with some parts of the environment):
   - this model shall be sensible and useful[5] [6];
   - this model shall show how the subsystems are decomposed in modules;
   - this model shall clearly clarify whether modules are SFR-enforcing, SFR-supporting or SFR-non-interfering.
2. The evaluator takes a sample of modules and explains the purpose for each sampled module and its interaction with other modules. This explanation shall, where possible, make use of examples from the developer evidence (e.g. diagrams).

Example of the module level model:

[5] That is, the modules should not correspond one-to-one with subsystems and they should provide a further level of detail than that provided for the subsystem; they should not just be a division of the subsystem with no additional explanation of the design of the security functionality.
[6] It is highly recommended that the evaluator presents a model that is (closely related to the model) used by the developer. The model presentation shall include references to the relevant sections of the developer evidence.



Example of the module behaviour and interaction (of the red module)



Result: The evaluator demonstrates that he understands the TOE design at module level and that all modules are identified and described.

4.3.2 The alternative ADV method for modules

In the alternative ADV method for modules, all requirements for the modules are met by the implementation representation.

As the implementation representation itself is considered to act as the documentation of the modules in the alternative ADV approach, all modes are implicitly categorized as SFR-enforcing. It is at the time of tracing SFRs to the implementation representation[7] (and hence also to modules and subsystems) that the evaluator makes a distinction between that which is SFR-enforcing and SFR-supporting and that which is SFR-non-interfering. If the evaluator traces an aspect of the implementation representation to SFRs, then it is considered to be SFR-enforcing or SFR-supporting, depending on the role the evaluator determines it plays in achieving the SFR. The evaluator can use their vast experience to quickly determine whether an aspect of the implementation representation does not play a role in achieving the SFR and hence is SFR-non-interfering.

Subsystems are sets of modules and the interfaces of those subsystems are the externally accessible interfaces of the modules. If the modules are sufficiently described, then also the subsystems are sufficiently described and additional subsystem level descriptions are not required.

[7] See section 4.4.2

While the full mapping needs to be completed in order to ensure the necessary modules are identified for ADV_TDS, there is no need to provide or present the full mapping of the SFRs to the modules. The presentation must only provide evidence by showing an example of how this mapping is generated and, on demand, the evaluator must be able to show how a specific SFR is implemented by the modules.

Therefore, only limited evaluator actions are required at this point.

The evaluator presents the method used to identify the modules from the sets of implementation representation (e.g. source code or VHDL), providing examples of the identified modules and rationale of how they fit the method for identifying modules (e.g. modules could be represented by source code classes, each source code function could represent a module).

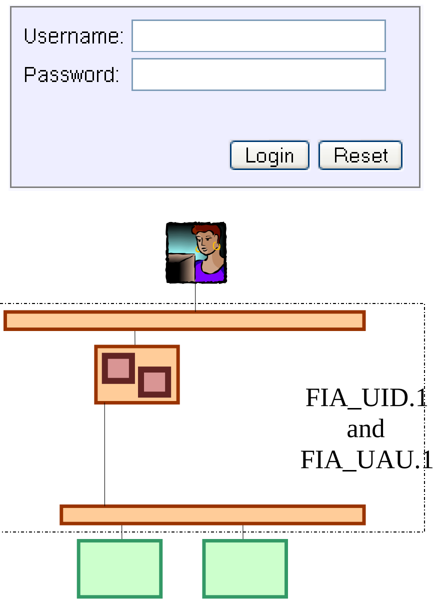
Result: The evaluator demonstrates that he understands the TOE design at module level and that all modules are identified and described.

4.4 Tracing SFRs to TSFI, Subsystems and Modules

4.4.1 The regular ADV method for tracing

1. The evaluator presents, for each SFR, how the TSFIs, subsystems (and modules) provide this SFR, using the TOE, diagrams, screenshots, submodules etc.
2. Where SFRs/TSFI interactions are complex (e.g. FMT_SMF applying to multiple administrator interfaces) this shall be split and clarified.
3. The evaluator describes what the role is of the TSFIs, subsystems (and modules) in meeting these SFRs.

Example of relation between SFRs, TSFI, subsystems and modules



Result: The evaluator demonstrates that he understands the TOE Design and FSP, and their completeness w.r.t. the SFRs.

4.4.2 The alternative ADV method for tracing

A mapping from SFRs to modules/subsystems is generated as a result of completing the ADV_TDS activities, as discussed in Section 4.3.2 above. Therefore, no additional mapping of SFRs to modules/subsystem is required here. However, the alternative ADV method still requires a mapping from the SFRs to the TSFI with references to the main points in the implementation representation as input.

It is important to observe that this information is extremely well suited for techniques such as pre-compiled evidence (i.e. cases where SFRs in Protection Profiles mandate compliance to an implementation standard so that SFR ~ TSFIs mappings are product independent). The reason is that product interfaces (TSFIs) are comparatively stable.

The evaluator uses his vast experience with the TOE type to identify all security relevant part of the implementation representation from these high-level starting points using classical implementation review techniques like data flow analysis, tracing of call chains etc.

This way, the evaluator ensures that all tracing requirements for the modules (and hence by inference, the subsystems) are met by the implementation representation without the need of explicit SFR-tracing information provided by the developer.

1. The evaluator maps each SFR to the relevant implementation representation items.
2. The evaluator presents a few examples of this mapping.
3. The evaluator must be able to describe for each SFR how it is realised in the implementation representation.

Result: The evaluator demonstrates that he understands the TOE Design and FSP, and their completeness w.r.t. the SFRs.

4.5 Security Architecture

This section applies to both the regular and the alternative ADV method.

The evaluator presents the security architecture and explains:

- how the TOE maintains security domains;
- how the TOE initialises;
- how the TOE protects itself from tampering;
- how the TOE prevents bypass.

This presentation will be targeted towards the model developed in the previous sections (i.e. consider subsystems and modules if applicable) and explains how the implemented security mechanisms contribute to the security properties.

When applying the alternative ADV method, reference to standard architecture for that TOE type (e.g. within a Protection Profile) should be made and used as a basis of the explanation of the main security features implemented in the implementation representation to meet the ADV_ARC requirements.

Again this is well suited to pre-compiled evidence techniques as the high-level security architecture concepts (as considered in ADV_ARC) are comparatively stable, and change only gradually over time.

Result: The evaluator demonstrates that the security properties are described and that he understands how they are achieved by the TOE,

4.6 Formal Security Modelling and other Formal Aspects of ADV (Optional)

In case the evaluation of the assurance component ADV_SPM.1 is in the scope of the evaluation, the evaluator adds the evaluation results to the ADV Presentation.

The evaluator includes the assessment for formal modelling aspects in other assurance classes (if claimed) in this part of the presentation as well.

Observe, that in the case that any formal modelling is claimed, the Alternative ADV Approach **cannot** be applied. This includes requirements such as ADV_TDS.6 as discussed above because the developer has to formally model their behavior and therefore also define the TSF subsystems in the developer evidence.

For ADV_SPM.1 “Security policy modelling” the evaluator presents that:

- the security policy is modelled in a formal style;
- all policies modelled define the security of the TOE and that the formal proof shows that TOE cannot reach an insecure state;
- the correspondence between the model and the functional specification is at the right level of formality;
- the functional specification is consistent and complete with respect to the model.

For ADV_FSP.6 “Complete semi-formal specification with additional formal specification” the evaluator presents the results of the assessment:

- of the formal specification of the TSFI supported by informal explanatory text where appropriate.

For ADV_TDS.6 “Complete semiformal modular design with formal high-level design presentation” the evaluation presents the results of the assessment of

- the formal specification of the TSF subsystems supported by informal explanatory text where appropriate
- the proof of correspondence between the formal specifications of the TSF subsystems and the functional specification.

Result: The evaluator demonstrates that the formal modelling, any associated proofs and explanatory text meet all the requirements, and that the formal specification is consistent with all other ADV evidence.

5 Implementation Representation Sampling Rationale

This section consists of three cases:

1. ADV_IMP.1 is used in conjunction with the regular ADV method
2. ADV_IMP.1 is used in conjunction with the alternative ADV method
3. ADV_IMP.2 is used in conjunction with either method.

5.1 The Sampling Rationale for ADV_IMP.1 with the Regular Method

This is a small presentation that describes the subset of the TOE Implementation Representation that will be examined and why this is assumed to be representative for the entire set. The actual evaluator work of ADV_IMP is handled in the TOE Implementation Representation presentation (see Chapter 8).

The evaluator shall present:

- the selected sample of implementation representation;
- a justification for the selected sample of implementation representation including the considerations that were given in this selection process.

Result: The evaluator demonstrates that he has chosen a proper set of Implementation Representation.

5.2 The Sampling Rationale for ADV_IMP.1 and the Alternative ADV Method

In case the alternative approach is used, the whole implementation representation is made available to the evaluator because it is required to gain the required information about the modular (and hence subsystem) design of the TOE. Therefore, no sampling rationale is necessary in the alternative ADV method for the EM1 deliverables.

The evaluator uses the implementation representation also to acquire the information about the modular design of the system. As a consequence, the correspondence between the modular design inferred by the evaluator and the implementation representation is implicit and no sampling rationale is needed.

There is nothing for the evaluator to present in relation to EM1 deliverables.

Result: The CB implicitly approves the sampling strategy as the whole source code is used by the evaluator in the ADV_TDS and ADV_IMP activities for the alternative ADV approach.

5.3 The Sampling Rationale for ADV_IMP.2 and the Alternative ADV Method

Since ADV_IMP.2 is used, the entire implementation representation is considered, and there is no sampling for the correspondence between the implementation representation and the design.

6 The ADV/AGD Reference Document

This document (not a presentation) is a list of references to the evidence, showing that certain ADV requirements are met that are hard to capture in a presentation. It consists of an ADV part and an AGD part.
The goal of the document is to show to the certifier *that* the work was done, but not give much detail on *how* it was done.

The certifier can perform spot checks if so desired. It is not intended that the certifier repeat part of the ADV or AGD evaluation by completely checking everything.

6.1 The ADV-part

1. The evaluator shall ensure that the ADV/AGD Reference Document contains detailed references (for each TSFI):
    - to the evidence where the parameters for that TSFI are described;
    - to the evidence where the actions are described;
    - to the evidence where the error messages and exceptions are described.
    - (for the discussion of non-TSFI error messages as required in higher ADV_FSP component-levels the evaluator can decide whether to present the results in the ADV/AGD Reference Document or in the TOE Implementation Representation Presentation)
2. The evaluator shall make available the relevant ADV documentation for spot checks during the meeting.

No example, as it is self-explanatory

Result: The evaluator demonstrates that all TSFI are fully described.

6.2 The AGD part

1. The evaluator shall ensure that the ADV/AGD Reference Document contains detailed references:
    - to the list of user roles;
    - to the list of user-accessible functions and privileges to be controlled in a secure processing environment (OPE.1.1C);
    - for each user role, how that user role is meant to use the available interfaces in a secure manner (OPE.1.2C);
    - for each role, the functions and interfaces available to that user role, plus parameters and values (OPE.1.3C);
    - for each role, the security relevant events (OPE.1.4C);
    - to the general description of modes of operation for the TOE, and how to maintain secure operation for each mode (OPE.1.5C);
    - to the security measures needed to fulfil each SO for the environment (OPE.1.6C);
    - to the acceptance steps (PRE.1.1C);
    - to the installation and preparation steps (PRE.1.2C).
2. The evaluator shall make available the relevant AGD documentation before the meeting.

No example, as it is self-explanatory

Result: The evaluator demonstrates that all AGD requirements are met.

7 The Configuration Item Identification Presentation

This is a relatively small presentation of a single ALC item: the identification of configuration items (as required by ALC_CMC.2/3/4/5.2C. The “Configuration Items” of interest are the identification means for all relevant parts / components of the TOE including their configuration like versioning information for all hardware and software components that constitute the TOE, and additional information like patch-levels, versions of configuration tables etc.

This is presented to allow the certifier to track how configurations items change when the TOE is patched as a result of testing.

In the first evaluation meeting, the evaluator must present for all Configuration Items listed in the ST (including the TOE and its guidance):

- What the identification (including version) of those Configuration Items is in the ST, and
- how would those identifications change if the Configuration Item changes (e.g. version number is increased, hash value changes, patch level is increased), and
- what method will the user and the evaluator use to verify these identifications (e.g. commands to send to the TOE and responses, comparison of hash values, comparing document identifiers and names). If these methods are different, both need to be clear and linked.

Even if no change to the Configuration Items is expected, it still must be clear how any changes would be visible from the identification.

The remainder of ALC is handled in the ALC presentation (see section 11).

The evaluator shall present the method used to uniquely identify the configuration items.

No example, as it is self-explanatory

Result: The evaluator demonstrates how configuration items are uniquely identified.

8 TOE Implementation Representation Presentation

This section consists of three cases:

1. ADV_IMP.1 is used in conjunction with the regular ADV method
2. ADV_IMP.1 is used in conjunction with the alternative ADV method
3. ADV_IMP.2 is used in conjunction with either method

8.1 ADV_IMP.1 is used in conjunction with the Regular ADV method

The evaluator shall present:

- Findings of implementation representation inspection, including the form of the implementation representation inspected.
- Any changes/additions to the (agreed) selected sample made as a result of the analysis. For example, where analysis of a selected portion of the implementation representation led to the inclusion of an additional area to clarify an ambiguity.

Result: The evaluator demonstrates that the selected portions of the implementation representation are consistent with the design.

8.2 ADV_IMP.1 used in conjunction with the Alternative ADV method

The evaluator shall present:

- Findings of implementation representation inspection, including the form of the implementation representation inspected.
- How the sample selection of SFRs is implemented in the implementation representation.

Result: The evaluator demonstrates that the implementation representation meets all SFRs, and, that as the implementation representation equals the design:

- the implementation representation is consistent with the design.
- the subsystems implement all SFRs.
- the modules implement all SFRs.

8.3 ADV_IMP.2 used in conjunction with either method

The evaluator shall present:

- Findings of implementation representation inspection, including the form of the implementation representation inspected.
- A mapping (in the form of a table) of all SFRs to the implementation representation.
- How the sample selection of SFRs is implemented in the implementation representation.

Result: The evaluator demonstrates that the implementation representation meets all SFRs, and, that as the implementation representation equals the design:

- the implementation representation is consistent with the design.
- the subsystems implement all SFRs.
- the modules implement all SFRs.

8.4 Presentation of TSF Internals (ADV_INT) (optional)

This section applies to both the regular and the alternative ADV method.

If one of the ADV_INT assurance components are claimed for the evaluation the evaluator presents the evaluation result as part of the TOE implementation representation in the second evaluation meeting.

The evaluator shall present:

- The criteria the developer used for well-structuredness and complexity of the TSF internals
- The results of the assessment of the well-structuredness and complexity of the TSF internals on the level required for the relevant assurance component. Under the regular ADV method this assessment is based on the developer internal analysis, which is then confirmed during the analysis of the implementation representation. Under the alternative ADV approach, this is based entirely on the evaluator findings during analysis of the implementation representation, which may be backed up by reports from static analysis tools.

Result: The evaluator shall report the criteria used by the developer and demonstrate that the well-structuredness and complexity requirements of the TSF internals are met.

9 The ATE/AVA Test Plan Presentation

9.1 Approach (overview)

The approach will consist of the following phases:

1. The evaluator will analyse the developer testing and creates an overview test plan.
2. The evaluator will present the developer testing and the overview test plan to the certifier. This will be done at the second evaluation meeting. The evaluator will distinguish between:
   a. Tests done by the developer which will be repeated by or witnessed by the evaluator;
   b. Tests done by the developer which will not be repeated or witnessed;
   c. Additional tests done by the evaluator;
   d. The rationale for choosing all of the above.
3. The evaluator will analyse all the other evidence and come up with a vulnerability analysis and penetration test plan based on this evidence.

9.2 Two methods for Developer ATE

For the evaluation (and presentation) of developer ATE under this NSP#6, there exist two methods:

1. The regular ATE method,
2. The alternative ATE method.

The alternative method for ATE is to be used in cases where the developer has a mature test system that can be used to show (near) completeness of developer ATE testing. The regular ATE method is to be used in all other cases.

In order for a laboratory to use the alternative ATE method, the CB must give permission. Therefore, the use of this method must be documented in the EWP.

With the alternative ATE method, the developer is able to provide a Developer Testing Rationale: a demonstration of the (near) completeness of testing by other means than explicit enumeration and mapping of tests to TSFI, subsystems and modules. This can include, but is not limited to:

- Tests suites that test against a given interface standard (e.g. the JavaCard standard);
- Tools that measure code coverage;
- Tools that systematically generate tests from code or interface specifications.

In this case, the evaluator can analyse the Developer Testing Rationale to establish that ATE_COV and ATE_DPT have been met, supported by sampling to determine that the Developer Testing Rationale is correct.

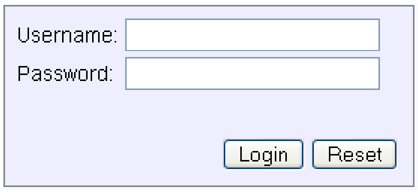
9.3 Coverage

9.3.1 Coverage under the regular ATE method

The evaluator shall present[8]:

- a systematic overview of which tests have been done by the developer;
- how these tests cover the various TSFIs.

Example of coverage



[8] This presentation may be integrated with the "Tracing SFRs to TSFI and Subsystems" presentation (Section 4.4).

TEST 1: Non-existent username
TEST 2: Incorrect password
TEST 3: Empty password
TEST 4: Correct password

Result: The evaluator demonstrates that all TSFI have been tested by the developer.

9.3.2 Coverage under the alternative ATE method

The evaluator shall present:

- The Developer Testing Rationale on why all TSFIs are tested;
- How he sampled the developer tests to determine that the Developer Testing Rationale was correct

Example of coverage

"*The developer uses the CodeComplete v4.18 tool to show that his tests have code coverage of 98.2%. The developer explained that the remaining 1.8% of the code, either:*

- *does not exhibit behaviour visible at an external interface, or*
- *represents errors that do not normally occur*

The evaluator sampled several functions from different places in the code and determined that these were tested by the test set of the developer. The evaluator also sampled:

- *some code to verify that it was not visible at the external interfaces*
- *represented errors that do not normally occur*

and found this to be the case."

Result: The evaluator demonstrates that all TSFI have been tested by the developer.

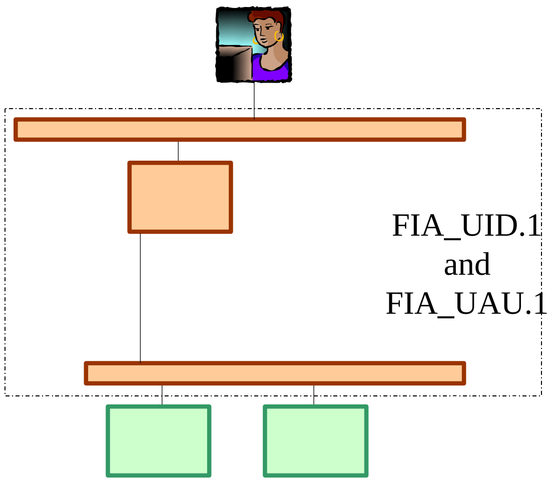
9.4 Depth

9.4.1 Depth under the regular ATE method

The evaluator shall present[9]:

- a systematic overview of which tests have been done by the developer;
- how these tests cover the various subsystems, modules or the implementation representation of the TSF (details depend on the ATE_DPT component level relevant of the evaluation)

Example of depth



TEST A: Performing login retrieves correct password from password file
TEST B: Performing login correctly compares entered password with stored password

Result: The evaluator demonstrates that all TSF subsystems have been tested by the developer.

9.4.2 Depth under the alternative ATE method

The evaluator shall present:

- The Developer Testing Rationale on why all subsystems (and modules / the TSF implementation depending on the chosen ATE_DPT level) are tested;
- How he sampled the developer tests to determine that the Developer Testing Rationale was correct

In many cases, the Developer Testing Rationale for subsystems (and for modules / for the implementation of the TSF) will be identical to or largely overlap the Developer Testing Rationale for TSFI. In that case, the presentation should be combined.

Result: The evaluator demonstrates that all subsystems (and modules / the TSF implementation) have been tested by the developer.

9.5 Developer Test Plan

The evaluator shall present:

- a sample of the test plan to show general style and how it meets the required criteria.

[9] This presentation may be integrated with the “Tracing SFRs to TSFI and Subsystems” presentation (Section 4.4).

Result: The evaluator demonstrates that the test documentation contains all necessary information. This is also demonstrated through the ability of the evaluator to repeat the selected sample of developer test cases.

9.6 Evaluator ATE Test Plan

The evaluator shall present[10]:

- the selection of developer tests that will be repeated;
- the additional evaluator tests.

Result: The evaluator demonstrates that he has chosen a proper set of ATE tests

The certifier is expected to comment on the two sets of tests during the second evaluation meeting, and the evaluator and certifier will come to an agreed ATE test plan.
If so desired, the certifier can indicate which tests he intends to witness.

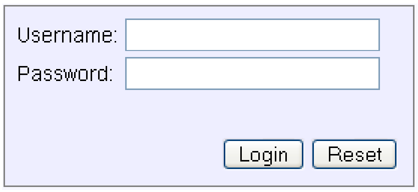
9.7 Evaluator AVA Test Plan

The evaluator shall present[11]:

- the results of the public domain vulnerability search;
- the focus of the independent vulnerability analysis (if applicable);
- the results of the independent vulnerability analysis (possibly supported by an additional Implementation Representation review report, see also Section 8);
- the resulting AVA tests.

Note that the evaluator should include argumentation in his presentation allowing the certifier to judge the completeness as required by the assurance requirements. Overview tables and consistent naming can support this significantly.

Example:



PENTEST 1: Standard accounts root/root, root/toor, anonymous/guest, guest/guest
PENTEST-2: Extremely long password
PENTEST-3: Password containing ^C, ^H and/or ^Z

Result: The evaluator demonstrates that he has chosen a proper set of AVA tests

The certifier is expected to comment on the search, analysis and AVA test plan during the second evaluation meeting, and the evaluator and certifier will come to an agreed AVA test plan.
If so desired, the certifier can indicate which tests he intends to witness.

[10] This presentation may be integrated with the “Tracing SFRs to TSFI and Subsystems” presentation (Section 4.4).
[11] This presentation may be integrated with the “Tracing SFRs to TSFI and Subsystems” presentation (Section 4.4).

10 The ATE/AVA Test descriptions

As the presentations for the ATE and AVA test plan will only present a very general test goal, the evaluator shall also deliver an ATE/AVA Test descriptions (this is a document).

The ATE/AVA Test descriptions shall contain:

- all tests of the ATE and AVA Test Plan Presentation
- for each tests, the objective, test method and expected result

Example:

Test 10: MD5 Signatures
The actual use of the md5 signature will be tested: tap NTP traffic and determine it uses the MD5 authentication properly.

- Objective: Establish that the ntp service is using password authentication so that an attacker cannot inject a false time into the TOE.
- Method:
  1. Record an NTP timestamp from the server
  2. Replay the ntp reply one hour later
  3. Check the time on the EMS server
- ExpRes: The time on the EMS server is not affected by the false reply

Result: The evaluator demonstrates that he knows how to execute the AVA and ATE tests

The certifier can sample this Test description for sufficiency. It is not intended that he completely verifies this document.

11 The ALC Presentation

The overall goal of ALC is for the evaluator to understand the processes and procedures applied in the TOE development and manufacturing lifecycle and to then gain confidence that the processes and procedures are applied as documented. This is a two stage process:

1. Review the documentation provided by the developer to understand the processes/procedures and to develop a plan of what is to be verified and how to verify the application.
2. Gain confidence of the application of the processes and procedures. Confidence may be obtained through site audit(s) or through evidence of their application (e.g. completed review documents, logs of access control mechanisms) provided by the developer.

The evaluator shall present:

- An overview of each ALC assurance family:
  - A summary of how the developer meets this family;
  - A summary of the evidence that the developer has provided.
- A checklist/plan of how to verify application of the processes and procedures.

The following items shall specifically be addressed:

- The life-cycle model, including the site(s) where development and production takes place;
- Physical, procedural, personnel and other security measures and why these measures are appropriate and sufficient for the TOE.

The evaluator shall make available the relevant STAR reports (if applicable) for spot checks during the meeting.

Result: The evaluator demonstrates that the developer meets the ALC Criteria and that the evaluator has a plan of how to verify the application of these measures.

11.1 Site Visits under this NSP

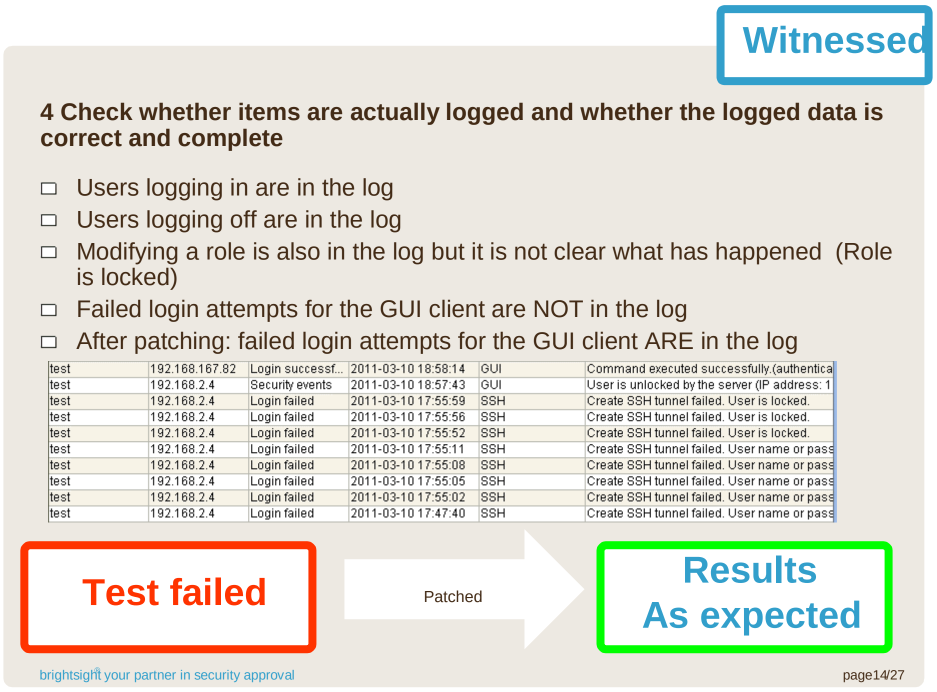
By default, NO site visits done for evaluations at EAL3 or below. However, this does not mean that no evidence of compliance with ALC should be gathered by the evaluator where ALC_DVS.1 is claimed but no site visit is performed: the evaluator should obtain evidence from the developer that he indeed follows the described procedure: screenshots of CM systems, photographs of physical security measures etc. Should the developer provide insufficient or confusing evidence, the evaluator and/or certifier may judge that a site visit is needed after all.

12 The ATE/AVA Test Results

The Evaluator shall present[12]:

- the test results of all tests in the ATE/AVA Test plan;
- if any tests failed, how these failures were handled by the developer and the test results of the subsequent evaluator retest.

Example of Test:



Witnessed

4 Check whether items are actually logged and whether the logged data is correct and complete

- ☐ Users logging in are in the log
- ☐ Users logging off are in the log
- ☐ Modifying a role is also in the log but it is not clear what has happened (Role is locked)
- ☐ Failed login attempts for the GUI client are NOT in the log
- ☐ After patching: failed login attempts for the GUI client ARE in the log

test	192.168.167.82	Login successf...	2011-03-10 18:58:14	GUI	Command executed successfully.(authentica
test	192.168.2.4	Security events	2011-03-10 18:57:43	GUI	User is unlocked by the server (IP address: 1
test	192.168.2.4	Login failed	2011-03-10 17:55:59	SSH	Create SSH tunnel failed. User is locked.
test	192.168.2.4	Login failed	2011-03-10 17:55:56	SSH	Create SSH tunnel failed. User is locked.
test	192.168.2.4	Login failed	2011-03-10 17:55:52	SSH	Create SSH tunnel failed. User is locked.
test	192.168.2.4	Login failed	2011-03-10 17:55:11	SSH	Create SSH tunnel failed. User name or pass
test	192.168.2.4	Login failed	2011-03-10 17:55:08	SSH	Create SSH tunnel failed. User name or pass
test	192.168.2.4	Login failed	2011-03-10 17:55:05	SSH	Create SSH tunnel failed. User name or pass
test	192.168.2.4	Login failed	2011-03-10 17:55:02	SSH	Create SSH tunnel failed. User name or pass
test	192.168.2.4	Login failed	2011-03-10 17:47:40	SSH	Create SSH tunnel failed. User name or pass

Test failed

Patched

Results
As expected

brightsight your partner in security approval

page14/27

Result: The evaluator demonstrates that the TOE has passed ATE and AVA tests.

[12] It is not intended that this consists of a set of “Pass”. Detailed descriptions and screendumps are to be provided where appropriate

13 The ALC Results

1. The evaluator shall present the results of the verification that the lifecycle processes and procedures are applied.
2. The evaluator shall provide a STAR report in accordance with the relevant requirements, if applicable and requested in the application form.

Result: The evaluator demonstrates that he has checked whether the developer applies the documented procedures.

14 Consultancy/Evaluation Improvement Presentation

Often, during the consultancy of an evaluation (or during the early stages of an evaluation) the developer makes significant security improvements to the TOE as the result of this consultancy/early evaluation. This process is often invisible to the certifier.

In some evaluations, when many or all of the problems have already been eliminated, the evaluation itself is a relatively sterile affair: the design is solid and all tests pass and it seems that both evaluator and certifier have contributed nothing to the security of the TOE.

To prevent this, this NSP mandates a Consultancy/Evaluation Improvement presentation.

The evaluator shall present:

- The security improvements made to the TOE during the consultancy phase. Note that this is only possible if the same lab also performed the consultancy. If this is not the case, this part of the presentation is skipped.
- The security improvements made to the TOE before the Evaluation Meeting, as a result of evaluation activities.

Example of improvements:
During the consultancy, it was noticed that:

- The TOE always used the same communication key
- The TOE was not resistant against SQL-injection

All of this was repaired before the evaluation started.

During the evaluation, it was noticed that:

- There was an “anonymous/guest” account
- The TOE did not log start and stop of the audit functionality

All of this was repaired before the First Evaluation Meeting.

Result: The certifier obtains insight in the security improvements of the TOE.

It should be noted that such information is only reported in the reports discussed in the evaluation meetings, and not in the final reporting (i.e. this information is not included in the ETR document (including any ETRfc) or in the Certification Report.

A. Example mapping of evaluator actions

The table below provides an example of how the evaluator might report the mapping of CC evaluator actions (to a level of content and presentation elements) for an EAL4 evaluation to the evaluation reports. The evaluator will populate such a table with the reference to the report(s), including details of the slide (in the case of a presentation report) or section number (in the case of a document) in which the action is reported.
Note that this table may need to be expanded with additional elements in case of composite evaluations.

CC Family	**Element**	***Report reference, including slide# or section #***
ADV_ARC1.1E	1.1C	
	1.2C	
	1.3C	
	1.4C	
	1.5C	
ADV_FSP.4.1E	4.1C	
	4.2C	
	4.3C	
	4.4C	
	4.5C	
	4.6C	
ADV_FSP.4.2E		
ADV_IMP.1.1E	1.1C	
	1.2C	
	1.3C	
ADV_TDS.3.1E	3.1C	
	3.2C	
	3.3C	
	3.4C	
	3.5C	
	3.6C	
	3.7C	
	3.8C	
	3.9C	
	3.10C	
ADV_TDS.3.2E		
AGD_OPE.1.1E	1.1C	
	1.2C	
	1.3C	
	1.4C	
	1.5C	
	1.6C	
	1.7C	
AGD_PRE.1.1E	1.1C	
	1.2C	
AGD_PRE.1.2E		
ALC_CMC.4.1E	4.1C	
	4.2C	
	4.3C	
	4.4C	
	4.5C	
	4.6C	
	4.7C	
	4.8C	
	4.9C	
	4.10C	

CC Family	Element	***Report reference, including slide# or section #***
ALC_CMS.4.1E	4.1C	
	4.2C	
	4.3C	
ALC_DEL.1.1E	1.1C	
ALC_DEL.1.2D (implied evaluator action)		
ALC_DVS.1.1E	1.1C	
ALC_DVS.1.2E		
ALC_LCD.1.1E	1.1C	
	1.2C	
ALC_TAT.1.1E	1.1C	
	1.2C	
	1.3C	
ATE_COV.2.1E	2.1C	
	2.2C	
ATE_DPT.1.1E	1.1C	
	1.2C	
ATE_FUN.1.1E	1.1C	
	1.2C	
	1.3C	
	1.4C	
ATE_IND.2.1E	2.1C	
	2.2C	
ATE_IND.2.2E		
ATE_IND.2.3E		
AVA_VAN.3.1E	3.1C	
AVA_VAN.3.2E		
AVA_VAN.3.3E		
AVA_VAN.3.4E		