

---

## UK IT SECURITY EVALUATION & CERTIFICATION SCHEME

UK CC Interpretation - UK/3.1/011

12 March 2007

**Status:** Endorsed for use in UK Scheme

**Subject:** Use of Open Source/Third Party Source Code

---

**REFERENCES** CC/CEM, Version 3.1, Revision 1, September 2006

### Issue

1. Open source code is being more widely used by commercial developers for integration into their products. It is a cost-effective alternative to implementation of bespoke code as open source code is readily and freely available. Examples of this are the open source BSD Unix operating system kernel, Apache webserver and MySQL.
2. Third party code is that which is developed by a third party software vendor and distributed to those software vendors wishing to integrate the third party code into their product, or integrate interface capabilities into their product. The distribution of this code is controlled to specified parties, usually under a formal contractual arrangement.
3. The ultimate conclusion of integrating open source code is the case where the TOE is entirely open source and the TOE sponsor's added value is limited to the packaging and validation of the open source code.
4. This interpretation addresses the question of how to treat open source or third party code that has been integrated into a TOE by the TOE developer.

### Scope of Interpretation

5. This interpretation is intended to identify any implications on evaluation activities according to the source of the implementation representation; whether developed by the TOE developer, a third party or an open source community.
6. Graduation of the interpretation will be based upon the assurance level/assurance package claimed for the TOE, limited to those assurance components included in EAL4 and the ALC\_FLR family.
7. This interpretation does not consider the instance where the TOE developer is not responsible for the delivery of the third party/open source item(s).
8. The integration of third party hardware and firmware items is not explicitly covered within this interpretation.

## Premises

9. This issue is based upon the following premises:
  - a. CC does not prohibit a TOE developer from making use of open source or third party code within a TOE;
  - b. The open source/third party code will be brought under the control of the TOE developer prior to distribution to the consumer.

## Interpretation

10. From the point of view of the evaluation, the main concern regarding third party/open source code is verifying that no Trojan code or backdoors have been introduced. Confidence in the changes made by the TOE developer is obtained through evaluation of the development procedures. However, where this is not possible for third party/open source code, the confidence has to be obtained through the developer acceptance procedures and inspection of the code.
11. There are three factors that may affect the treatment of source code integrated in a TOE developed by a party other than the TOE developer:
  - a. The evaluation assurance level/package - *graduation of assurance level*.
  - b. A third party, with whom the TOE developer has a contractual agreement and support for the code, develops the source code or the code is developed by an organisation/individual(s) with no obligation to the TOE developer for support - *originating organisation*.
  - c. The criticality of portions of the TOE implementation comprised of code developed outside the TOE development organisation - *criticality of third party/open source*.

## Originating Organisation

12. If the source code is provided by an organisation with which the TOE developer has contractual agreement and support, the TOE developer may opt to impose requirements on the third party to provide information regarding the third party development processes and procedures.
13. This option would mean that the TOE developer would be able to present configuration management and security procedures for the third party's development sites as well as their own, rather than levy additional requirements on the TOE developer's acceptance procedures to verify integrity of the source code received from the third party.
14. The provision of third party development procedures would enable a view of how security of the source code is maintained during development. It would, however, introduce a requirement for the secure transmission of source code items between the third party and the TOE developer. Also, where the assurance requirements specify evidence be provided to demonstrate the application of processes and procedures in the development environment, evidence should be obtained from both the TOE developer and the third party development sites.
15. This approach is not possible without the full co-operation of the originating organisation, and is therefore not typically an option for open source code.

16. The scenario of the TOE sponsor's main added value being packaging and distribution of the TOE is likely to only occur where the code is obtained from an open source, rather than a commercial third party. In this instance the evaluation evidence and evaluator activities relating to ALC should focus on the checks made by the TOE sponsor when taking control of the code to verify its integrity (e.g. ALC\_CMC.4-10).

#### Criticality of Third Party/Open Source

17. The criticality of open source code can increase the rigour the evaluator will apply to the assurance requirements interpreted above. If the source code developed by an open source provides part of the TSP-enforcing functionality of the TSF, a greater level of rigour will be applied to the consideration of that source code than open source code in the TSF not providing TSP-enforcing functionality.
18. If ADV\_IMP.1 is included in the assurance package, in addition to the factors normally applied to selecting the source code sample, the criticality of the code developed by the open source should be factored into the selection of the sample. However, the increased attention to the TSP-enforcing code produced by the open source may result in a sample of source code that is larger than that selected if the developer has had control of the source code through-out its lifecycle. This is reflected in the following section.
19. This approach could be used equally where the source code is obtained from a commercial third party, although in this instance the TOE sponsor has the option to impose contractual requirements upon the third party organisation to provide the required development environment evidence, as documented in the previous section.

#### Graduation of Assurance Level

20. When considering the issue of the *graduation of assurance level*, the question arises of when the origin of the implementation becomes a consideration in the evaluation. At lower levels of assurance the control of the source code development and the actual development is not visible to the evaluator. It is not until the development procedure evidence includes items such as acceptance of new/modified configuration items into configuration management and the evaluator is provided with the source code and the development security procedures that the evaluator should take the origin of the code into consideration.
21. Therefore, in terms of the EAL assurance packages, it is not until EAL4 that the origin of the code must be considered during an evaluation. Attention to this issue is required to gain confidence, not only in the correct implementation of the TOE against the specification, but also that the TOE has *only* been developed against the specification; ensuring no Trojan code or backdoors have been inserted.
22. Each CC Part 3 class is discussed in turn below, identifying where components are affected by the origin of source code items. The following also takes into account the other factors detailed in paragraph 10, where applicable.

#### *ADV activities*

23. Representations of the TSF for any third party/open source components are to be

provided as for all other portions of the TSF.

24. If ADV\_IMP.1 is included in the assurance package, a sample of source code to be inspected must be selected. In addition to the factors documented in CEM paragraph 1822, the selected source code sample should include a sample of each type of third party/open source code present in the TOE:
  - i. Pure third party/open source code, as received;
  - ii. Third party/open source that has been the subject of minor modifications by the TOE developers (e.g. changing of names to be consistent with developers naming convention, application of minor patch/bug fix) ;
  - iii. Third party/open source that has been substantially re-written by the TOE developers.
25. In the instances where the entire TOE is comprised of third party/open source code, the evaluators are to use a static analyser to parse the code to verify the correct structure (i.e. closure of loops, use of all defined global variables, etc). The output of the analyser will be subjected to sampling by the evaluator, forming part of the implementation representation sample selected by the evaluators. The evaluator will also sample the source code itself for manual inspection to consider the implementation of key security functionality, applying the sampling guidance provided in [CEM] Annex A.2.
26. If the third party/open source code has been substantially re-written by the developer, then in support of the major concern of the introduction of backdoors/Trojan code during third party/open source development, the focus of manual evaluator inspection should be on those items of code that have not been modified by the developer and therefore more likely to contain backdoors/Trojan code.
27. Where the TOE has been sourced from third party/open source code and the developer has only made minor modifications, the code will be examined as for those TOEs comprised entirely of third party/open source code.

#### *AGD activities*

28. The installation/secure operation of any user interfaces of third party/open source code components of the TOE, visible to the TOE user, are to be described in a manner commensurate to that for all other portions of the TOE.

#### *ALC activities*

##### *ALC\_CMC/CMS*

29. The open source/third party source code will be accepted as a (set of) configuration item(s) at a given point during the TOE development.
30. The identification and acceptance of third party/open source items will need to be considered under ALC\_CMC.

ALC_CMC.1	The origin of configuration items is not evident at this level. Therefore, no explicit consideration of these items is required.
ALC_CMC.2	The labelling of any third party/open source code should be consistent with the described method of labelling, as considered under ALC_CMC.2.3C and the associated work unit ALC_CMC.2-4 (the

	<p>identification of source code items on a configuration list will be dependent upon the granularity of the configuration list). Therefore, the origin of code is not expected to be visible.</p>
ALC_CMC.3	<p>As for ALC_CMC.2, once the third party/open source item is controlled under the developer CM system, the integrity of the item is maintained in the same manner as any other source code item under the control of the system. Therefore, again the origin of code is not expected to be explicitly visible.</p> <p>In addition, ALC_CMC.3 considers authorised changes to CIs. At this level the origin of the CI is not visible; it is just a CI maintained within the CM system. Therefore, no explicit consideration of these items is required and authorisation of changes to third party/open source code should be consistent with that for all other implementation representation CIs.</p>
ALC_CMC.4	<p>In addition to ALC_CMC.3, the acceptance of third party/open source configuration items is to be addressed in the CM plan, as considered in ALC_CMC.4.8C and the associated work unit ALC_CMC.4-10.</p> <p>Where the third party/open source code <b>does not</b> provide TSP-enforcing functionality, it is sufficient to examine the procedures to accept new/modified CIs to determine that externally produced items are appropriately considered prior to acceptance as (part of) TOE configuration item(s).</p> <p>However, in the instances where the third party/open source code <b>does</b> provide TSP-enforcing functionality, the procedure to accept new/modified CIs should be sampled to verify it's effectiveness in ensuring integrity of the item(s). This should include inspection of evidence generated through application of the procedure, e.g. test results and code inspection sign-off.</p> <p>Now that the source of CIs is visible through the procedures to accept new/modified CIs this should be considered in the authorisation of changes to CIs. If changes are made by the TOE developers to items previously developed by an external party, these changes should be considered as any other change made to the TOE implementation representation. If the external party makes further changes, the submission of the modified items should be considered to be new items and subject to the procedure to accept new/modified CIs.</p>

31. There is no impact on ALC\_CMS requirements, as any items used in the construction of the TOE will be considered within the implementation representation items.

32.

*ALC\_DEL*

33. As stated in the Scope of Interpretation, it is assumed that the TOE developer is responsible for the delivery of the third party/open source items. Therefore, the delivery procedures should consider the delivery of these items as for the remainder of the TOE.

### *ALC\_DVS*

34. It is also not necessary to consider the third party/open source developer security procedures when the acceptance criteria of the items is sufficient to gain confidence in the secure implementation. This confidence may be gained through activities such as:
- a. peer review of open source items;
  - b. a thorough testing and source code inspection of the item by the TOE developer, distinct from the ATE testing activities, which are focused on the provision of security functionality rather than the absence of Trojan code/backdoors;
  - c. modification of the items by the TOE developer to the extent where all portions of the item relied upon for the provision of TSP-enforcing functions of the TOE have been revised.
35. However, where third party items are bought from another developer under a contractual agreement, the TOE developer may chose to specify the third party development site as an extension of their own development environment, submitting the developer security procedures of the third party for assessment.

### *ALC\_LCD*

36. The lifecycle model presented to satisfy the ALC\_LCD.1 requirements should include the development of portions of the TOE by any third party/open source. If ALC\_CMC.4 is included within the assurance package, this information should be consistent with the acceptance criteria described in the acceptance procedures. The lifecycle model is to consider the following aspects of third party/open source components:
- a. identification of need for inclusion of items;
  - b. identification and acceptance of items;
  - c. integration of items within the TOE;
  - d. further development (maintenance) of items.
37. In the instance that third party/open source items integrated into the TOE are not providing TSP-enforcing functions, the security procedures employed during the development of these third party/open source items do not need to be considered during the TOE evaluation. For example, an open source report generation tool may be integrated into a TOE to generate reports of network load-balancing if the TOE is a network appliance. The generation of these load balancing reports may not be required to meet any SFRs and therefore do not form part of the TSF and are not relied upon by the TSF.

### *ALC\_TAT*

38. The tools and techniques employed for the development of source code by a third party/open source should be considered to the extent required to support evaluator activities. i.e. the language in which the third party/open source code was developed and any design tools should be reported by the TOE developer to support the evaluator's ADV activities. The associated versions of tools should be reported.
39. Peer review of code within the open source community is the accepted mechanism for identifying the use of any questionable constructs within the source code item. E.g. kernel modules developed in open source communities are subject to stringent peer

review and regression testing before being 'released'. Therefore, although it is not possible to definitely assert that a coding standard has been applied without considerable detailed inspection of the code, commensurate assurance can be gained that general good coding practises have been adhered to through the peer review.

#### *ALC\_FLR*

40. The remediation of flaws identified in third party/open source items must be carefully considered in the light of the satisfaction of other assurance requirements. Consider the example where BSD Unix is integrated within the TOE. Following acceptance of the open source BSD Unix source code items, a bug is found in the version of BSD Unix on which the TOE is based. A patch is developed and issued to address the bug by the open source community. The TOE developer determines that the bug is relevant to the TOE. The developer has substantially modified the implementation of BSD Unix, and so it has been determined that consideration of the security procedures applied during the development of BSD Unix do not need to be considered. Therefore, to simply apply a fix issued to address the bug may undermine the confidence in the development procedures previously gained for the BSD Unix source code.
41. This example highlights the need to consider the criteria for acceptance of bug fixes and patches in the configuration management system.

#### *ATE activities*

42. The tests sub-activities are to be performed as for all other portions of the TOE.

#### *AVA activities*

43. The vulnerability assessment sub-activities are to be performed as for all other portions of the TOE. Therefore, analysis of vulnerabilities should take account of vulnerabilities reported in the third party/open source items as well as those reported for the TOE. Any such vulnerability should be analysed to determine applicability to the TOE, taking into consideration modifications made to the implementation during integration with the TOE, as well as consideration of the TOE threat environment and operational environment. This applies to AVA\_VAN.1, AVA\_VAN.2 and AVA\_VAN.3.
44. The applicability of potential vulnerabilities arising from the use of third party/open source items is easier to identify for AVA\_VAN.3 activities than AVA\_VAN.1 and AVA\_VAN.2 due to the other assurance components included in the assurance package.

### **Rationale**

45. This interpretation is consistent with CEM paragraph 60, which states that assurance requirements apply to the whole TOE, but did not elaborate by providing guidance of the implication on development procedures where aspects of the TOE were provided by third party/open source items.
46. The CEM assumes development of the TOE by a single development organisation, as implied by the reference to developer in the singular (CEM 8.3.2) and guidance for the ALC activities (as provided in CEM 13), with the exception of the mention in ALC\_CMC.4-10 that items may be accepted from other manufacturers. This

interpretation assumes that there is a single developer responsible for the development, integration with any third party/open source items and distribution of the TOE, while recognising that the TOE developer is likely to reuse third party/open source code where possible.