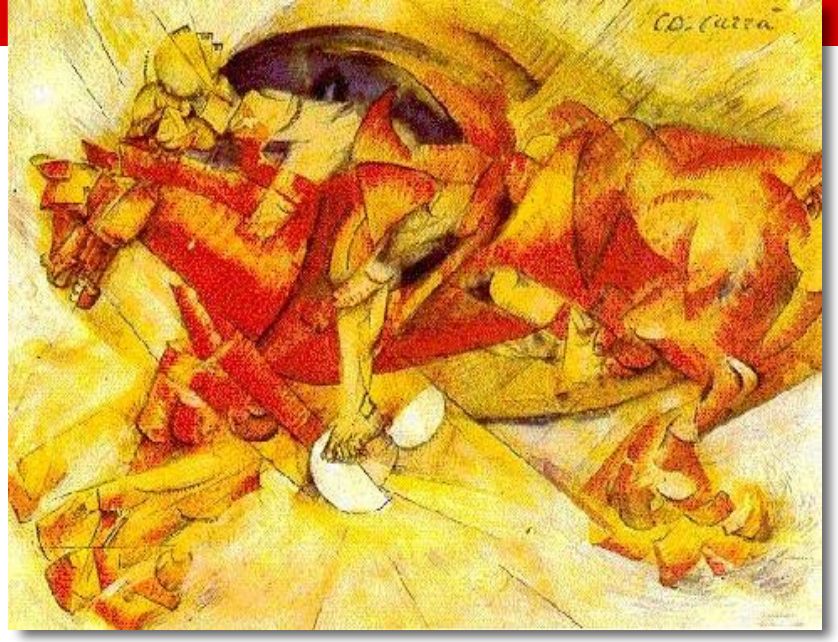
the information security provider



Evaluating Third-Party Code:
How Can It Be Trusted?

Courtney Cavness

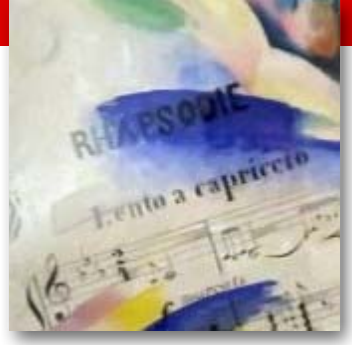
www.atsec.com
courtney@atsec.com

Overview

- Purpose of “process“ work units
  - establish trust in the TOE developer
- Purpose of “technical“ work units
  - validate that the TOE security functionality works as expected
- **Problem**: Incorporated, third-party component
  - No defined method in the CC to establish trustworthiness of the third-party hardware/software developer or the third-party code itself
- **Solution**:
  - Proposed “Trusted Supplier“ assurance package from EAL2 to EAL4
  - Defined acceptance procedure at EAL3 and EAL4

> At lower levels, this issue is not evaluated. For higher assurance levels, the requirements expressed in this paper may not be sufficient, and more formal analysis of the semantics of third-party code and more strict controls of the development environment may be required to ensure that an attacker with a high attack potential would be unable to attack the organization’s computers and manipulate the code developed on its systems.

ALC “Process” Families:



- CMC / CMS – reliable version control of TOE items is in place
- DEL – a secure delivery process is used
- DVS – sufficient logical and physical access controls are applied to protect the code from unauthorized access, and hiring practices help ensure employees are themselves trustworthy
- FLR – flaw reports are accepted and tracked, and consumers are alerted to fixes
- LCD – lifecycle processes are in place (i.e., to facilitate approval of code advancing through development/test/release stages)
- TAT – known and dependable tools and processes are used to create and implement the TOE

All help establish that the developer is a “trusted“ entity.

Trusted Developer

The TOE developer demonstrates that they:

- Have “vetted“ employees (via some established hiring practices)

  Note: Currently, CC evaluation doesn‘t ensure the developer or it‘s employees are non-criminal (i.e., there is no investigation into company shareholders / validation of specific results of background checks, if performed).
- Protect the TOE from unauthorized access
- Can be depended upon to create a reproducible end product (i.e., version control)
- Offer support for flaws, so the product remains stable in the future
- Provide secure delivery (preventing masquerading attempts and ensuring the end product is authentic)

This provides assurance that the TOE developer‘s product is not malicious, its integrity is maintained, and it is authentic.

”Technical” Classes:

- ADV
  - validate that the TOE security functionality is sufficiently documented (per EAL)
- ATE
  - validate that the TOE security functionality is adequately tested (works as expected / claimed)
- AVA
  - validate that there are no vulnerabilities in the TSF (per EAL, either publicly-known or discovered via pen test)

All help establish that the TOE code itself is trustworthy.

Problem …

In many cases, the developer undergoing evaluation did not create all TOE components, but incorporated some from third parties.

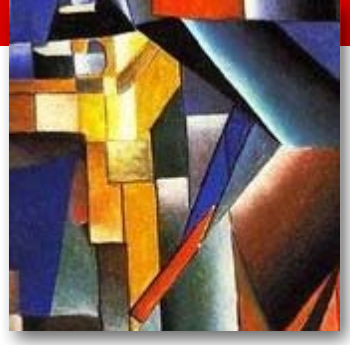
- Public libraries
- Applications
- Kernels
- Operating Systems
- Code inherited through mergers and acquisitions
- Flaw fixes

...if any of these are created by a different organization, they are considered third-party components

Some organizations don‘t even realize they have incorporated third-party components in their products.[1]

[1] *“Report: Reused, Third Party Code Major Sources of Insecurity,“*
30% of applications submitted to Veracode as “internally developed“ contained code from third party suppliers.
http://threatpost.com/en_us/print/6737

Third-Party Components: Benefits



- Bundle applications for end-user convenience
- Avoid development efforts spent “re-inventing the wheel“
- Save money by using more cost-efficient labor
- Take advantage of different time zones for quick turn-around
  - Example: testing/flaw fix performed on newly-developed code in a cycle not interrupted by typical business hours in a single country

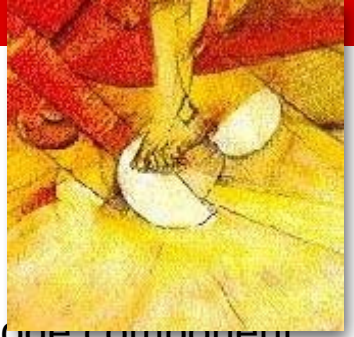
Third-Party Components: Drawbacks

- The component might inadvertently introduce vulnerabilities:
  - Some flaws occur *simply as an extension of being incorporated with another developer's code*.
  - Insufficient parameter checking

    For example, one developer has a module that has no external interfaces, and assumes that trusted interfaces won't request invalid parameters. However, another component developer has an external interface that does not check for valid parameters because it merely passes it on to the other component, potentially exposing the code to buffer overflows, cross-site scripting, and similar types of vulnerabilities, etc.
- Documentation errors (open ports, etc.)
- The code/component might not be able to be fully understood
  - Lack of source code, documentation, design information, testing
- Configuration errors
- The component could contain back doors or be malicious

A developer could be held legally liable if their products contain malware.

Malicious Code: What is the Potential?



Sample Attack Scenario: A logic bomb incorporated in an open source code component that "phones home" with sensitive information gathered.

Using the table on page 421 of the CEM, the author's estimation of the attack potential shows that the corresponding acceptance procedure should be EAL3 or EAL4.

Factor	Least Amount	Most Amount
Time	2	4
Expertise	6	6
Knowledge of TOE	0	3
Window of Opportunity	4	10 (assuming remote access is possible to internal network)
Equipment	0	0
RESULTS:	12 – Enhanced / Basic (EAL3)	23 – High (EAL5)

Types of Third-party Developers

- Unknown (and therefore, inherently untrusted)
  - Open Source
  - Unwilling to be evaluated
  - Unable to be evaluated (i.e, inherited, “old age“ code)

- Known
  - Can be identified and contacted
  - Willing to undergo evaluation for trustworthiness

How to Trust Each Type

- Known third-party:

  SOLUTION: Evaluate the third-party developer, not the component

  "Trusted Supplier" (SAR package) separate evaluation performed by accredited laboratory

- Unknown (inherently untrusted) third-party:

  SOLUTION: Evaluate the component, not the third-party developer

  Acceptance Procedure performed by the TOE developer, verified during TOE evaluation.

  Currently, the CC contains the concept that an acceptance procedure should be performed at EAL4 for third-party code (ALC_CMC.4-10). However, such a procedure is not defined, nor are any specific requirements provided.

In both cases, determining whether the security functionality (if any) provided by the third-party code works as expected, is adequately tested, and matches the ST claims can be verified by an accredited laboratory during the "technical" portion of CC evaluation.

Proposed “Trusted Supplier” Assurance Package

- The Common Criteria already provides the framework necessary to establish the trustworthiness of a developer: the ALC “Process” families.

- Code from these Trusted Suppliers could then be incorporated into other IT products and treated no differently than if it were produced by a TOE developer’s own resources; it would simply undergo their normal integrity and functionality tests since the following would have already been established about the third-party developer:
  - lifecycle processes are in place (i.e., to facilitate code peer-review and approval of code changes)
  - hiring practices check the background of their employees
  - flaw reports are accepted and consumers are alerted to flaw fixes
  - a secure delivery process is used
  - version control is maintained
  - sufficient logical and physical access controls are applied to protect the code from unauthorized access

Question: Who pays for third-party evaluation?

- TOE Sponser: may specify that the TOE include the third-party component.
- Third-party developer: market differentatior, can easily be included in an IT developer‘s supply chain.
- TOE developer: will save money / reduce development effort by using the third-party.

Trusted Supplier Methodology

At a minimum, the following are required to establish the third-party developer as “trusted“ (per slide #5):

- CMC.2 (uses versioning – unique ID)
- CMS.3 (maintain the implementation representation itself - not object code - *for supply chain considerations*)
- DEL.1
- DVS.1 (add mandatory non-criminal background check of employees required)
- FLR.3 (to “push“ AUTOMATIC distribution of fixes to registered users)
- LCD.1 (add mandatory peer-code review to internal best practice standards)
- TAT.1 (since the TOE developer may not accept compiled code and integration of a component developed in a different language is an issue)

Current ALC Assurance Class

Family	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
CMC	1	2	3	4	4	5	5
CMS	1	2	3	4	5	5	5
DEL		1	1	1	1	1	1
DVS			1	1	1	2	2
FLR							
LCD			1	1	1	1	2
TAT				1	2	3	3

Trusted Supplier Methodology (Cont'd)

- Certification valid for 2 years and tied to the specific site(s) listed on the Trusted Supplier certificate.
  - Initial site visit required, then sites re-visited only if there is substantial change in physical security (i.e., building change or merger resulting in different physical security applied). In lieu of continual site visits, other evidence such as internal audit results (for adherence to processes), developer interviews via phone, digital photos (of minor physical security changes) will suffice.

- Trusted Supplier certification can also apply to other CC efforts **within the same organization**. In other words, if a global company was undergoing CC certification of a product developed/tested etc. in the US and Canada, and both pertinant sites had up-to-date Trusted Supplier certification, no ALC evaluation need be done.

  Note: Trusted Supplier evaluation covers CMS for the types of items required to be maintained in the CM system. It won't be examined by consuming TOE developers.

- EAL of the Trusted Supplier must match (or be higher) than the TOE being evaluated. Default EAL level is equivilant to EAL2, but TOE developer can augment only the CMC and/or CMS families to upgrade to EAL3 or EAL4. *Does not apply to EAL5 or higher.*

- If different tools, CM systems, lifecycle models, delivery methods, etc. are in use at a site, they must **all** be evaluated.

Trusted Supplier: Related Concepts

This proposed methodology supplements and augments the related concept:

- The Bundesamt für Sicherheit in der Informationstechnik (BSI) Site Certification
  - Similarities
  - Differences

Evaluation approach for “unknown” third-party code:

Problems are likely:

- Incompatibility error
  - Found as part of normal TOE developer testing or CC evaluation activities.
- Design / logic error
  - May be found as part of CC evaluation activities (AVA / IND)
  - May be found during manual source code review
  - Requires design documentation from TOE developer regarding:
    - purpose of the third-party code
    - how the third-party code is used in the TOE

      Note: Evaluated in ADV work units if the third-party code implements TSF or has dependencies to the TSF. Examined in AVA work units if does not affect the TSF, but might still cause vulnerabilities (i.e., XSS in the presentation layer of a web-based software but not part of “business logic“.) If 3rd-party code can interfere with TSF (i.e., running in the same process space), it can be considered the TOE environment.
- Malicious code
  - May be found during manual source code review

    Note that functional tests are a “positive“ check on whether the expected functionality is provided, but a “negative“ check is necessary to make sure it doesn‘t perform additonal, unexpected functionality. This type of negative check is performed during AVA and manual code review.

Proposed Acceptance Procedure

- Which code is subject to acceptance procedures?

  All third-party code that is part of the TOE security functionality (TSF) must undergo acceptance procedures, regardless if it is SFR-enforcing, SFR-supporting, or SFR-non-interfering.

  Note: For code that is not reviewed, it must be obvious that any - even malicious - behavior of this code will have no impact on the enforcement of the security objectives and the SFRs defined in the Security Target.

Proposed Acceptance Procedure (Cont'd)

- ALC_CMC.4-10 - must be expanded to define an acceptance procedure at EAL4 to consist of a documented, manual review (potentially combined with some tool-based analysis) performed by one or more of the TOE developer's vetted employees.

  The review should be performed on any TSF code that is developed or changed by subcontractors or third-parties (including open source communities) to make sure it performs the expected — and only the expected — functionality. The review should be especially looking for malicious code, unintentional vulnerabilities, back doors, etc.

  - If the incorporated code is large, the effort to perform code review is a daunting task.
  - Note that running code through software such as FxCop and/or Coverity is can be used to support the acceptance procedure (i.e. to minimize bugs) but the tools themselves cannot interpret what the code is doing.

- ALC_CMC.3-10 must be added to define an acceptance procedure at EAL3 to consist of TOE developer unit test specifically for the third-party TSF code.

Proposed Acceptance Procedure (Cont'd)

- ALC_CMS - The developer must keep a copy of the incorporated third-party implementation representation at the specific version for both EAL3 and EAL4.
  - The 3rd party code accepted into the TOE must be the actual source files. Object code that is pre-compiled by a 3rd party cannot be used to build the TOE, because that code cannot be evaluated, effectively rendering it "black box." There is no way to relate compiled .jars (object code) back to source code.
  - ALC_CMS.2-3 should be expanded:
    - Current: The evaluator shall check that the configuration list indicates the developer of each TSF relevant configuration item, if different from the TOE developer undergoing evaluation. For example, the relevant third-party component developer or subcontractor.

Proposed Acceptance Procedure (Cont'd)

- ALC_FLR is required. The TOE developer must assume the responsibility for monitoring for flaw reports for the third-party code, obtaining/creating flaw fixes, testing the fix, and securely distributing the fix to the TOE users. This is true for not only incorporated, but also bundled, third-party code.
- AVA should specify that the evaluator must review any incorporated third-party code for potential vulnerabilities (TSF or not).
- TDS design documents and test coverage of the third-party code must be provided by the TOE developer.

Proposed EAL3 Acceptance Procedure (Cont'd)

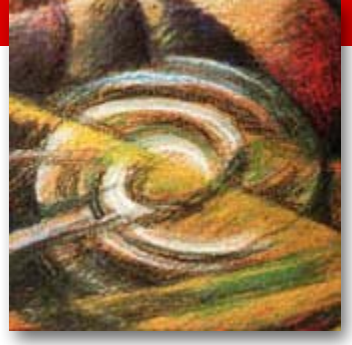
- ALC_LCD – The lifecycle information must include a role responsible for determining when the third-party code is finalized, and how and when it is integrated into the code base, and which roles are allowed to make additional changes after that point.
  - The third-party developer can even check the source code into the TOE organization's CM system, but it is the TOE organization that must decide which code is included in the product build.
  - Only the code that gets integrated into the build is of concern. After integration, the code is under the control of the TOE organization and remains so, and the CC evaluator is only concerned with what happens from that point in the life cycle forward.

Proposed EAL3 Acceptance Procedure (Cont'd)

- ALC_LCD – The acceptance procedure should be documented to describe:
  - role/department that reviews the code
  - depth/rigor of review (description of activities/tools)
  - lifecycle of how/when files are reviewed, accepted, and checked into the respective CM system for maintenance / inclusion into the TOE.
  - how it is determined that the 3rd-party has finished its work?
  - How the review activities are documented (and that document itself, managed)

- ALC_LCD.1-1 term "subcontractor" CEM para 1181 should be expanded to include third-party code and trusted suppliers.

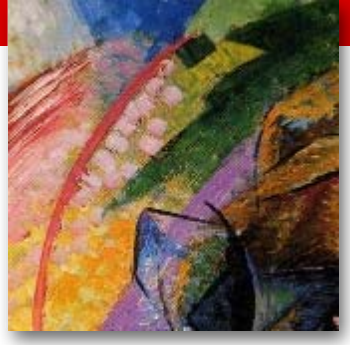
Summary



- Incorporating third-party components implies a certain level of committment to that component.
  - The difference between being involved and being committed:

    Bacon and Egg breakfast

  - The developer who incorporates a third-party component is committed to testing / maintaining it. It is now the TOE developer‘s reputation, liability, future sales at stake.
- The CC already contains the means to establish “trust“ in third-party developers.
  - Existing CC work units can be arranged in a separate assurance package to certify a known third-party developer as “trusted.”
  - In situations where the third-party is unknown, expanding and specifying the already-existing CC notion of an “Acceptance Criteria” can offer some assurance that the code itself is “ trusted.”



Acknowledgements:

I would like to thank my valued colleagues who gave their expert opinions on what constitutes a valid acceptance procedure, and provided input / review of this presentation and the ideas contained within:

- Helmut Kurth
- Stephan Mueller
- Fiona Pattinson
- Yi Mao
- Clemens Wittinger
- Jeremy Powell
- Sebastian Mayer