COMBINED SAFETY AND SECURITY CERTIFICATION

G. Romanski

Verocel, Inc., USA, Romanski@verocel.com

Keywords: MILS, Certification, Safety, Security.

Abstract

New systems are being developed which are used in safety critical systems but must also satisfy security requirements. To reduce space, weight and power, a Multiple Independent Layered Security (MILS) platform could be used to support many applications. A MILS platform could support safety critical, and non-safety related functions, and manage Top-Secret, Secret, and Unclassified data, by providing the necessary protection and controls to manage them. The certification of such a platform must organize a lot of interrelated data that both safety and security domain auditors find acceptable. The approach used to capture and manage the certification evidence to satisfy both safety and security properties is presented.

1 Introduction

The certification guidance documents, for safety and security state that software that is linked together and loaded into a single address space, may be certified independently of other components. In DO-178C [5] Data and Control Coupling must be verified by test to ensure that the links between code and links between code and data are interpreted correctly by the target hardware.

To reduce the number of possible links, and to help isolate potential problems to smaller components, the safety and security industry has moved to modular components. These components are mapped to robustly partitioned systems that integrate the components in a way that isolates their behavior so that it is only visible through published and agreed interfaces. Components cannot affect each other or be affected by each other except through the published and agreed interfaces that are controlled by the system integrator. These interactions are not limited to data, but they also include timing interactions, and interactions through shared resources.

Over the last 10 years most of the new transport aircraft developed include Integrated Modular Avionics (IMA) systems. The most commonly used specification for IMA systems is ARINC 653 [1]. These IMA platforms are capable of supporting many applications and provide an abstraction layer that controls collaboration between applications, fault management and interfaces to sensors and effectors. With the increase in processing power of the platforms, the applications have evolved to take advantage of the computational power offered and they have become more sophisticated and more complex. This has led to an increase in the amount of certification data that must be developed.

Over the last 5 years in addition to IMA platforms, Multiple Independent Layered Security (MILS) architectures have started to be used. These architectures provide robust and secure partitioning to allow software applications at different and independent security levels to coexist on the same platform. The applications are loaded by the system integrators onto a secure MILS platform. This platform is configured to allow the applications to provide their services but at the same time to prevent them from violating the security policies established by the system integrator.

As new systems are developed, more integration means that safety critical applications may process data that may have various levels of security. Some applications may be compromised if they are penetrated by external attacks that deny service or corrupt data. Others may need to protect data from being leaked to users without appropriate security clearance. Flight plans for covert operations must be protected yet flight plans must be filed with air traffic controllers to coordinate safe flight in non-segregated air space. Control of secure information is becoming more important with the increase in automation. This is particularly relevant in the remote control of unmanned air systems from ground-based stations.

The emergent solution for such systems are MILS architectures that support both Safety and Security based applications. These must be constructed well and support both safe and secure partitioning with a secure load and configuration system. The advantage of such systems is that they can support incremental certification where components can be assessed independently for safety, and for security. Each partition becomes a separate security domain that is unable to obtain or disclose information except through authorized channels. This means that analysis of an application running in a top-secret partition need not be evaluated with a high level of rigor, as it will be unable to send information directly or indirectly to a partition with a lower security level.

2 Process Driven Certification

Building a certification package for a security/safety critical system must be done in accordance with a set of process plans. DO-178C does not prescribe the process plans content or structure, but it expects them to be developed and reviewed as an integrated set. Together the plans provide a rigorous definition of what is to be done. Certification auditors will review the process plans and their use as described in section 9. The Common Criteria [2] provides a very broad description of the activities and objectives to be satisfied for the security domain. For a MILS type architecture, this is condensed to a description of a Separation Kernel Protection Profile (SKPP)[4] that is used to describe the attributes of a Secure Target (ST). Note that security certification of the MILS kernel focuses on the software architecture. The hardware architecture is the subject of a separate certification effort. The software certification assumes that the hardware performs as specified. Tests verify the correct behavior of the software running as a complete integrated image on the target hardware with no instrumentation of the code while under test.

Both security and safety verification work use process plans that describe activities to be performed, the criteria that must be satisfied before an activity is to start, and the criteria that confirm the completion of an activity. During the development of the certification materials, Quality Assurance personnel perform in-process checks, and end-of-phase checks. The in-process checks confirm that the documented versions of the process plans are being used, and that they are being followed. The end-of-phase checks verify that the set of activities representing each phase of verification are complete.

3 Requirements as Artifacts

Certification evidence is based on artifacts. The starting point is always the requirements, as these describe the intended behavior of the system and the software. DO-178C identifies requirements into one of two categories, high-level requirements and low-level requirements. High-level requirements are those that describe the "black-box" behavior of a system or software and they might not map directly to the architecture of the software. Low-level requirements are hierarchical, and follow the hierarchical structure of the software to which they correspond. Relationships are recorded between the requirements that capture various types of hierarchical or behavioral dependencies.

It is common practice in industry to capture the requirements in a database driven repository. The common practice is to extract the requirements as a whole or in sets and to baseline them in documents so that they can controlled and reviewed. The baseline identifiers are then used to identify the requirements of the system that will be built and verified.

An alternative approach is to apply a version identification to each requirement individually and to maintain lifecycle status over each requirement individually. A requirement may consist of a textual statement together with supplemental context information and various attributes. The requirement may be comprised of diagrams, tables, or other representations, or may simply reference such representations. All of the requirement information must held under Configuration Management (CM) control with direct and automatic links from the database to the CM repository or in the requirements repository itself. Version control of the individual requirements must extend to the version control in the CM representations so that a known and consistent set of requirements is always maintained.

Through the use of this finer granularity of control over the requirements, they can be processed through the review cycles individually or in small sets using attributes that control their states. A requirement will be in the Initial state when it is first introduced, either through import from another source or creation directly in the database. When the engineer responsible for them deems one or more requirements complete, they are transitioned to a *Ready-for-review* state. An independent engineer will review a requirement and mark it *Passed* or *Failed*. *Failed* requirements will transition back to the responsible engineer who may modify the requirement, change the version, and transition it back to a *Ready-for-review* state. *Passed* requirements will have an associated checklist that captures the criteria satisfied during requirement review and the history of the requirement review process. By increasing the granularity with which the requirement are managed, and by controlling this through an automated repository, the workflow can be shortened. It is no longer necessary to wait until an entire requirements document is ready for review, the reviews may be performed individually as soon as each requirement is ready.

Even though the certification guidance documents demand evidence of final review, the certification auditors will look for evidence that the review cycle was carried out in accordance with the plans. The evidence that shows requirements that failed review that were subsequently corrected and re-reviewed is important. These incremental steps must not be omitted from the certification package. The requirements presented for review must be identified by versions of the documents in which they were captured or by versions of the individual requirements that were reviewed individually.

4 Types of Requirements

During the development and certification of a MILS platform there are no System Level requirements. The platform itself is a component that will become part of a system when populated with application software. The applications will be based on System Level requirements. Several other requirement sets are defined during a typical MILS certification, for example:

- Requirements describing the Application Programming Interface (API) layer provide a specification that describes the behavior of the MILS

system. These are a subset of the high-level requirements.

- Requirements describing the secure behavior provided by the MILS kernel. These requirements are used for penetration testing, a form of testing which tries to induce security failures.
- Robust partitioning requirements are developed from a Robust Partitioning Analysis. A Goal Structuring Notation (GSN)[3] set of diagrams are prepared and analyzed. The diagrams capture the analysis performed and document the goals of robust partitioning by the MILS Kernel. The goals are traced to requirements at various levels of detail and ultimately trace to solutions that reference requirements and certification evidence directly. GSN was used for robust partitioning analysis because many of the vulnerabilities due to robust partitioning led to negative requirements. For example, "a partition shall not be able to change the memory of another partition". Such requirements are difficult to verify directly so an indirect approach was chosen based on the use of GSN.
- High-level requirements that describe the black box behavior of the MILS system, including the interface as presented to the board support package.
- Low-level requirements correspond to the actual software. They are structured and are map to Directories, Files, and Functions. At each of these levels more detailed requirements may be added to capture finer grained behavior.

Relationships are established between requirements. Hierarchical relationships are simple with simple references, but more complex relationships also exist.

5 Requirements Traceability

It is common practice in industry to represent traceability between requirements through the use of tables. One-to-one, one-to-many, and many-to-one relationships are shown on rows of a table with appropriate groupings. Other, document style representations are also used, where a document lists sets of requirements in sequence. An alternative is to capture the requirements as artifacts in a database and to capture the relationships between requirements as artifacts in the database as well. The requirement relationship artifacts will have version, status and review information attached to them and they represent the traceability between the requirements. A high-level security requirement may map to several lower-level requirements that in turn trace to the code that implements the expected behavior. There may be a significant difference between the abstraction levels of the requirements and an explanation or rational for the traces must be provided and reviewed with independence. The hierarchical low-level requirement traceability typically corresponds to the structure of the software. The traceability from high-level to low-level is typically complex and requires a high level of domain expertise. The traceability may be demonstrated by hyperlinking from one requirement to the trace justification data to the corresponding traced requirement, and presenting this information for review using a standard browser.

DO-178C does not describe the type of information to be provided in the traceability between requirements, but in practice auditors will ask for it if they do not understand the relationships. In the security domain Common Criteria (CC) v2.3 part 3 paragraph 314 states:

> *"... the developer provide evidence , for each adjacent pair of TSF (Trusted Security Function) representations, that all relevant security functionality of the more abstract TSF representation is refined in the less abstract TSF representation. ..."*

By recording these relationships, their versions and their review status, it is possible to automate impact analysis between requirements. A change to a low-level functional requirement may propagate to a high-level security requirement that may change its state which forces the requirement to be re-evaluated. This propagation of the impact due to changes includes the requirements and the explanation or rational they are associated with.

The alternative approach using documents to manage change control over traceability means, annotating and revising documents followed by a document review cycle. For large sets of requirements, such changes would need to be performed in batches. This makes the process unwieldy and stretches the intervals between review cycles, delaying the project.

5 Other artifacts

The software development and verification also included many other artifacts including Design descriptions, Source Code, Test Specifications, Tests, Coverage Analysis and so on. These artifacts are linked to the requirement and to each other as described in the planning documents. The artifacts are maintained in CM, but links in the requirements database should reference these artifacts directly.

Two approaches are used for version management. One approach identifies complete sets of artifacts and baselines these sets calling them versions. An individual file is not versioned, instead the file version is tied to the baseline. This is often used during a development process where the software components are released in complete software builds, where all of the latest file versions of the files comprise the version built.

For verification an alternative may be used, where each artifact is identified and is individually versioned, with lifecycle status information just like the requirements in their database. The use of individual file version control rather

than baseline control, increases the granularity over the control over individual artifacts.

The review process for the artifacts could be exactly the same as for requirements, except that different review criteria would be used. Review checklists may be captured in the database and may be extracted and checked by QA personnel, or presented to the certification auditors.

Through the use of a fine-grained database and requirements repository, impact analysis may be automated, and localized as much as the traceability links allow. A change to the source code of a function may cause a function level low-level requirement to be changed, but the low-level requirements of other functions in the same source file may be unaffected. Only the affected requirements and the artifacts that are dependent on them would have a state change in the database requiring the artifacts to be revisited by reviewers.

This fine granularity imposed on the artifacts makes concurrent working efficient because it reduces information that must be re-reviewed, and makes revisions less expensive through the use of automation. Automation also makes the verification processes more efficient by reducing the clerical errors that are often made when maintaining this information manually in documents.

6 Formal and Semiformal Methods

The SKPP requires use of formal methods to analyze information flow and to demonstrate that the Separation Kernel does not permit unauthorized information flow, and protects itself when adversaries attempt to compromise its behavior.

This is accomplished by translating the source code to a form that lends itself to analysis using automated theorem provers. A separate organization performed work on the formal proofs. The database was extended so that the requirements for the proofs were linked to the theorems, the source code, the extracted data flows and the proof results.

Special analysis is sometimes necessary as the objectives of safety and security sometimes clash. Simply stopping the operational system could mitigate a security failure while a safety critical failure may require special recovery actions. Managing requirements and analyses while satisfying safety and security properties is most complex during the management of failure conditions. If an application fails, a safety critical system identifies, isolates and instigates preplanned recovery functions and records the failure through a health management system. This must be accomplished without disturbing any other application on the same platform. If a security based application fails, the platforms responsibilities are to isolate the application, prevent the propagation of unauthorized dataflow, and to inject a security violation event to the security audit log.

Modeling such failure conditions formally is difficult especially as they are asynchronous. A semiformal approach using GSN has been used to capture and analyze the intended behavior. The resulting arguments are presented as evidence that the Separation Kernel (the core of the MILS system) preserves it safety and security properties in the presence of asynchronous events.

The use of data-flows used for the capture of requirements intended to test the Data/Control coupling objectives was also used as a semiformal method for the identification data paths that may be exploited as covert channels.

7 Maintaining the Database

A busy verification company will have many projects and many database repositories being used concurrently. Some companies arrange for database super-users to perform all official recording and update of requirements in each repository. A better approach is to provide everyone on the project controlled-access to the database they are working on. As work is often performed in several locations concurrently, it is important to solve a performance problem due to the network bandwidth. A single consistent database is required for a project. A single physical database could be used, but for any project other than the very small, network bandwidth will always be a problem. A solution that should be adopted is to use replicated databases that give everyone access to a single logical database that is synchronized automatically.

As each artifact is individually versioned, and the status of each artifact is known at all times, following the documented processes is enforced by the database. A design description cannot be officially reviewed until its requirements have passed review. Similarly for test cases; they cannot be officially reviewed until the requirements they are verifying are reviewed. These lifecycle relationships and dependencies are built-in and enforced in accordance with the documented process plans. By increasing the level of granularity of the artifacts and maintain their traceability, it is possible to work in parallel as much as the database allows, knowing that the order of verification steps is in accordance with the process plans.

The alternative approaches of managing the verification processes in batches, requires much more manual management and intervention.

8 Final Delivery

At the completion of the project, a large amount of data is available in the database and in the CM system. The traditional approach is to extract the documents, and cross check all of the information being packaged together. This is a tedious and error prone process that requires minute attention to detail.

An alternative is to use the information present in the database and CM repository. The extraction of this may be

automated. It could simply be accomplished by traversing a specified database, pulling the artifacts that correspond to a nominated variant using a baseline set. Information in the database could be formatted using specified style sheets and referenced information in the CM system could be fetched and saved in a set of directories in the file store. As the information in the database "knows" how to access all CM files, it is possible to generate a set of files that can be viewed by a generic browser. Traceability in the database can be converted to hyperlinked files that represent the "threads" of traceability an auditor is going to inspect. Picking a Security requirement, and following a link to an API level requirement, may be performed by clicking on the traceability link. This link also exposes the rationale for the trace, the reviewers, and review evidence. A click on the API level requirement will trace to the low-level requirements, their review history, design, source code, test specifications etc.

9 Certification Audits

The certification authority or their representatives perform audits at several points of the certification process. For safety critical certification audits, they typically impose four SOI's (Stage of Involvement). SOI#1 is performed when the planning stage is complete, SOI#2 is performed when at least fifty percent the design and code development and review is complete, SOI#3 is performed when at least fifty percent of the testing is complete and SO#4 is the final audit. The final audit examines the package presented.

The security audits are not specified with the same rigor, but progress on preparation of the security evidence is examines with equivalent care and attention.

The purpose of an audit is to obtain confidence that the entire certification package is complete and sound. It takes a long time to develop certification packages, and the auditors do not have the time or resources to perform a complete review of all artifacts. To compensate for this they rely on three principles:

1. The process plans for the certification work must be complete and they must be sound. This is accomplished through reviews early on in the project.
2. The results of using the process plans must be complete and sound. This is examined by taking several "threads" starting from highest-level requirements and tracing through all related artifacts and their evidence of review.
3. The quality assurance records are examined to verify that the process plans were being used consistently throughout the project and that all phases of the project have been completed.

The assumption is then made that if the "threads" examined are sound, and the approved processes were followed and completed then all other "threads" would also be sound. Once confidence is established the authorities are prepared to sign the audit reports.

The auditors for safety and security domains are typically highly experienced and very knowledgeable in their own domains. While they understand the needs and goals of their counterparts, they do not have the same level of knowledge. It is incumbent on the certification applicant and the verifiers to bridge the gaps during these audits.

10 Conclusions

The certification evidence for safety and security based operating system could be performed in parallel as the same evidence can satisfy many of the properties of the two domains. As is typical in software engineering projects, development, testing, certification for both safety and security may be performed concurrently. By maintaining fine-grained control, individually versioning requirements and all artifacts, and by maintaining the state of all database artifacts the impact of the software revisions can be isolated as much as possible. This reduces the rework necessary when software is 'improved'. It also maximizes workflow. This overlap between the safety and security domains should be exploited wherever possible, as much of the information developed for the safety domain helps in the security domain. Some of the analyses needs domain specific information to be captured and managed. The integration of the safety and security requirement into a single database, and the integration of this with CM repositories allow the automation of traceability. The automated traceability provide a huge benefit to the management and productivity on a MILS safety and security certification project.

References

[1] ARINC, *"Avionics Application Software Standard, Part 1 Required Services"* ARINC 653-2, December 1, 2005.

[2] ISO/IEC, *"Common Criteria for Information Technology Security Evaluation, Part 3 Security Assurance Requirements"*, ISO/IEC 15408, August 2005.

[3] Kelly, T. P. and Weaver, R. A. *"The Goal Structuring Notation – A Safety Argument Notation,"* Proceedings of Dependable Systems and Networks 2004, Workshop on Assurance Cases, July 2004.

[4] National Security Agency - USA, *"US Government Protection Profile for Separation Kernels in Environments Requiring High Robustness"*, SKPP Version 1.03, June 29, 2007

[5] RTCA, *"Software Considerations in Airborne Systems and Equipment Certification"* DO-178C, December 13, 2011