

COMBINING SECURITY AND SAFETY PRINCIPLES IN PRACTICE

T.J Cockram, S.R. Lautieri

Praxis High Integrity System, trevor.cockram@praxis-his.com, samantha.lautieri@praxis-his.com

Keywords: Safety, Argument, Security, Accreditation.

Abstract

In this paper we present an example of applying a combination of security and safety principles. The Ministry of Defence have been developing a common methodology for security accreditation and safety assurance within the SafSec project [5]. The example described in the paper applies this approach at a detailed level, using aspects of security to support the safety argument and safety techniques to support security accreditation. We show an argument, which uses the dependability by contract approach, and how this is used.

1 Introduction

Security and Safety have often been seen as separate and sometimes-conflicting disciplines but have significant overlaps [1]. The potential to combine safety and security at a process level has been researched investigating a combined approach to support accreditation and an acceptable safety case [2]. The first stage of the process is to identify both safety and security risks through a systematic process, and then to identify suitable control measures. At this stage conflicts and gaps between requirements can be identified and resolved. A common approach can then be made in developing arguments to support security accreditation and safety assurance, resulting in an improved process that reduces both technical and programme risk.

In addition to improving processes we can also exploit some common solutions to support the development of a product but applying security techniques in supporting safety and vice versa.

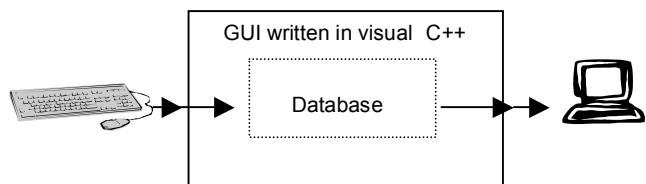
The remainder of this paper discusses the practical applications of these techniques through an example system.

2 System Description

The example we are considering in this paper is based on a sanitised version of an actual system that is part of a command and control system, which allocates the deployment of personnel within the battle-space.

The purpose of the system is to authenticate the user as someone who is permitted to enter configuration data into the system. The user manually enters the permitted and

prohibited locations, for personnel deployment, into a database. This high integrity data is validated as part the input process. Another authenticated user may input planning information, which is processed with the configuration data to support the Command and Control function. The result data then needs to be validated against the original input, containing the permitted and prohibited locations for the personnel. If the plan falls within the prohibited location then a critical accident may be the consequence and if the plan falls out side of the permitted location then a less serious accident may result. Therefore the validation of this data is a safety related function. The resulting plan also needs to be security controlled and may only be outputted by an authenticated user.



There are some constraints in that the computer hardware is COTS as is the operating system and that that the GUI is reused from another system and is written in Visual C++.

3 System Requirements

The original input data if credibly corrupted will result in a hazard, which without mitigation is not tolerable. The mitigation is in two parts, one on the input side confirming that the data is valid. We can use a physiological technique to validate the data as it is inputted. This is retrieved and displays a securely stored copy of the input data in a slightly different format to the input causing the operator to see errors before confirming the input. The second safety related function is to check that the processed data is valid against the original. As this application resides on a commercial off the shelf operating system we cannot claim credit for any part of its processing. The only security feature of the operating system that relates to the argument is that it controls user access to sensitive data. In addition to the normal accounts and passwords normally provided, the database carries out authentication of users.

The checking of the processed data against the original for conflicts and out of range values uses an approach that derives from the security domain. The data when original

stored is encrypted and provided with a security signature – assuming for the sake of argument that the db ‘owns’ the key. When the data is read back to compare with the processed data the security signature is checked. If data had been corrupted then the security signature will be invalid and the decrypted data will not be credible.

The safety target for the system is SIL 3. By this approach we have reduced the amount of safety related software by excluding all the GUI (SIL 2) and Operating System (SIL 1) and by separating the data validation from the command and control processing it may be possible to reduce the required integrity of the Command and Control software. This approach is known as “White Box Safety”. [3]

4 Module Boundary Contract

4.1 Module Boundary Contract

We use the SafSec approach to identify the module boundary contracts within the specification. We look at the five components of the contract: the guarantee clause; rely clause; context clause; assurance level; and counter evidence. The contract specification forms the basis of the argument for the required assurance.

The five components of the module’s boundary contract are described as:

- The guarantee clause specifies the conditions or dependability characteristics that the module guarantees to be true;
- The rely clause specifies the conditions or dependability characteristics, and their level of assurance, that the module relies upon from other modules in order to achieve it’s guarantee clause;
- The context clause defines the assumptions relating to the operational context of the implementation of the system components that contribute to the guarantee and rely clause;
- The assurance requirements (AR) clause defines the level of confidence or assurance that is claimed for the contract and can be for the module as a whole or against individual/sets of guarantees; and,
- The counter-evidence clause defines the limitations that exist in the contract such as known defects, residual risk etc.

In accordance with SafSec principles the example system architecture breaks down into a number of modules: physical environment; display unit; operating system; GUI; login; and, database (db). Each module is a logical grouping of dependability specifications for the purposes of abstraction and encapsulation.

Suggested partial module boundary contracts for the system’s identified modules’ boundary contract are provided in the following tables. The naming convention for the identifier within each module’s clause is *module.clause.type.number*.

Physical Environment Module Boundary Contract		
ID	Guarantee	AR
E.G.1	The physical environment shall be protected by physical security to prevent removal of the data input application	Serious breaches < 1 per year
ID	Context	
E.C.1	The physical environment is an MoD facility covered by all the usual policies and procedures that MoD facilities must follow.	
Display Unit Module Boundary Contract		
ID	Guarantee	AR
DU.G.1	The Display Unit shall guarantee to display data it receives correctly	Fails < 1x10 ⁻⁴ per use
ID	Rely	AR
DU.R.1	The example application must provide a Login and GUI as dictated by permissions.	SIL 2
DU.R.2	The physical environment must be protected by physical security to prevent physical removal of the tactical processor.	Serious breaches < 2 per year
ID	Context	
DU.A.1	The Display Unit will be used in an MoD facility.	
Operating System Module Boundary contract		
ID	Guarantee	AR
O.G.1	The Operating System shall guarantee to transmit message data correctly when requested	SIL 1

GUI Module Boundary Contract

ID	Guarantee	AR
G.G.1	The GUI shall guarantee to authenticate the user ID and password	SIL2
G.G.2	The GUI shall guarantee to accept a user ID and password in xxxx data type	SIL2
G.G.3	The GUI shall guarantee to provide access to the db if authentication is successful	SIL2
G.G.4	The GUI shall guarantee to send an error message if the user ID and password aren't authenticated	SIL2
G.G.5	The GUI shall guarantee not to corrupt the data entered for processing	SIL2
G.G.6	The GUI shall guarantee to send the data entered for processing to the db	SIL2
G.G.7	The GUI shall guarantee to display the data as entered by the user	SIL2
G.G.8	The GUI shall guarantee to provide confirmation/non-confirmation of the data entered for processing as valid	SIL2

ID	Rely	AR
----	------	----

G.R.1	The Login must not corrupt the data entered by the user	SIL2
G.R.2	The Login must send a user ID and password in xxxx data type	SIL2
G.R.3	The db must inform the GUI if the user ID and password are not held on the db	SIL2
G.R.4	The db must inform the GUI if the user ID and password are held on the db	SIL2
G.R.5	The db must provide access to the db if the user ID and password are authenticated	SIL2
G.R.6	The Login must display unsuccessful access to the db	SIL2
G.R.7	The db must indicate the data has been updated	SIL2
G.R.8	The Display Unit must guarantee to display data it receives correctly	Fails < 1x10 ⁻⁴ per use
G.R.9	The db must be able to accept the	SIL2

data sent to it for processing

G.R.10 The db must provide a signature on the data when confirmed to the GUI SIL2

G.R.11 The db must indicate the data is not valid and has not been updated SIL2

ID	Context
----	---------

D.C.1 The Login 'window' is not the same 'window' as the GUI 'window'

Login Module Boundary Contract

ID	Guarantee	AR
L.G.1	The Login shall guarantee to send a user ID and password to the GUI	SIL2
L.G.2	The Login shall guarantee to send the data as entered by the user.	SIL2
L.G.3	The Login shall guarantee to inform the user if the user ID and password is not in xxxx data type.	SIL2
L.G.4	The Login shall guarantee to display a unsuccessful attempt to access the db	SIL2
L.G.5	The Login shall guarantee to provide a graphical user interface	SIL2

ID	Rely	AR
----	------	----

L.R.1 The Display Unit must guarantee to display data it receives correctly Fails < 1x10⁻⁴ per use

L.R.2 The Operating System must guarantee to transmit message data correctly when requested SIL1

L.R.3 The GUI must guarantee to accept a user ID and password in xxxx data type SIL2

Database Module Boundary Contract		
ID	Guarantee	AR
D.G.1	The db shall guarantee to check for validity of the user ID and password	SIL2
D.G.2	The db shall guarantee to indicate the validity of the user ID and password	SIL2
D.G.3	The db shall guarantee to provide access to the existing data if the user ID and password are valid	SIL2
D.G.4	The db shall guarantee to provide all existing data with a signature	SIL2
D.G.5	The db shall guarantee to compare the data provided for processing with the existing entries in the db	SIL2
D.G.6	The db shall guarantee to provide confirmation/non-confirmation of the validity of the data entered for processing	SIL2
D.G.7	The db shall guarantee to indicate the existing data has been updated with the data for processing	SIL2
ID	Rely	AR
D.R.1	The GUI must be able to indicate to the user the data has been updated	SIL2
D.R.2	The GUI must be able to indicate to the user the validity of the user ID and password	SIL2
D.R.3	The GUI must be able to indicate to the user the validity of the data entered for processing	SIL2

4.2 Module Composition

Module composition ensures that modules are consistent with each other. A number of properties must hold true of the composed Module Boundary Contracts for the composition of modules to be possible, for example satisfaction arguments must indicate how the guarantees and relies of interfacing modules are satisfactory and adequate. Further properties are detailed in [5].

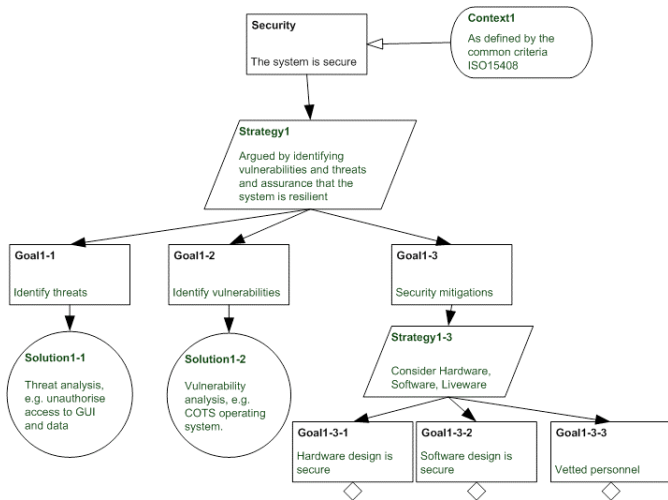
Examples of module composition using the modules defined above are detailed below. The tables are module centric and provide the guarantee with the relies required for the guarantee to hold true in the left column, while in the right column are the guarantees, in the interfacing modules, that satisfy the relies. For example, DU.G.1 requires DU.R.1 and DU.R.2 for DU.G.1 to be true, and the relies are satisfied by L.G.5, G.G.3 and E.G.1.

Display Module Composition	
DU.G.1	L.G.5
DU.R.1	G.G.3
DU.R.2	E.G.1
Login Module Composition	
L.G.1	O.G.1
	G.G.2
L.G.2	DU.G.1
L.R.1	
L.G.3	DU.G.1
Database Module Composition	
D.G.2	D.G.3
D.R.2	D.G.4
D.G.6	G.G.8
D.R.3	
D.G.7	G.G.8
D.R.1	

5 Argument

Conventional graphical argument techniques such as goal structuring notation appear to support the integration of safety and security arguments making clear the contribution of the work into the approval process, but in order to apply the process effectively the GSN syntax [7] and semantics require some simple extensions [4].

There are effectively three principle threads to the argument, one addressing the identification of security treats and vulnerabilities and the mitigations in place to reduce the risk of a security breach, and second thread relates to arguing that the safety of the system has been adequately addressed, and finally the certification/accreditation process to argue that the system is fit to enter service based of the evidence of the requirements verification, validation and relevant assurance processes. This fits with the generic SafSec argument structure [5] of losses identified, the requirements mitigate losses and the requirements correctly implemented

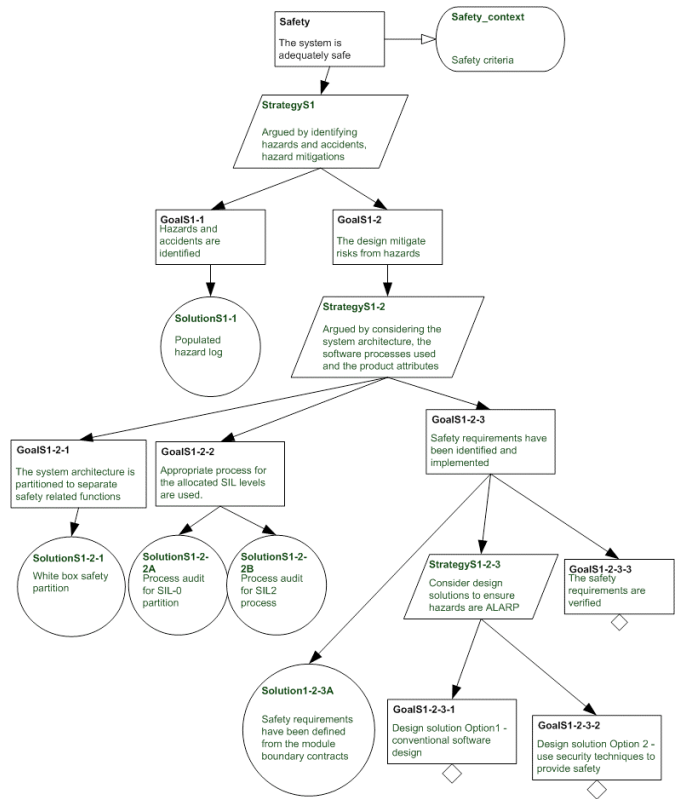


Starting with the security argument, taking the IT common criteria definition ISO15408 we argue that the system vulnerabilities and threats are identified and that that these are adequately mitigated to reduce the risk of a security breach.

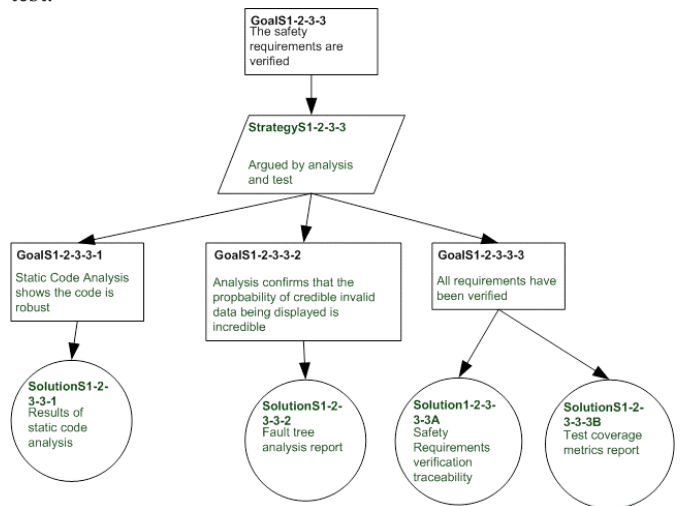
The safety argument is more conventional, but it should be noted that common techniques for identifying hazards and accidents can also be applied to identify threats and vulnerabilities, further information on exploiting this common approach can be obtained from the SafSec resources [5].

We make the argument concerning the system architecture here, but in fact it could be applied under Goal 1-3-2 above in the security tree, and in general; terms a system partitioning argument may help in the security assurance process, but is more appropriately contained within the safety argument.

This level of argument considering the design options is fairly superficial. This can be expanded to address the selection of options as described in [4].



Expanding on GoalS1-2-3-3 we consider both analysis and test.



The final component of the argument is that of assurance. From as safety perspective the use of an independent safety auditor (ISA) has been established [6]. In the integrated safety and security view we can use the concept of the ISSA (Independent Safety and Security Audit) here we use the similar expertise applied to safety in the security domain to ensure that the system is in a state for certification/ accreditation.

Conclusions

We have shown that there are benefits in some systems in combining safety and security principles. Each discipline has something to learn from the other. Module boundary contracts provide the process of specifying both types of attributes. Goal structure notation is an appropriate method to demonstrate both safety and security arguments. Similar work on module boundaries has been carried out by the IAWG and York University [8].

Acknowledgements

We would like to thank Dr David Jackson for reviewing the paper and Mr Brian Dobbing and Mr David Cooper for generating some of the ideas implemented in the paper.

Appendix A: More information on Module Composition

Additional example of module composition using the GUI module:

GUI Module Composition	
G.G.1	L.G.2
G.R.1	L.G.3
G.R.2	D.G.6
G.R.3	
G.R.4	
G.G.3	D.G.3
G.R.5	
G.G.4	D.G.6
G.R.3	L.G.4
G.R.6	
G.G.6	D.G.5
G.R.9	
G.G.7	DU.G.1
G.R.8	
G.G.8	D.G.7
G.R.7	D.G.4
G.R.11	D.G.6
G.R.12	

In addition to composing modules, systems can also be composed. When composing systems the accompanying dependability cases have to be consistent by establishing that the external dependencies and context clauses of one system on another system do not conflict.

References

- [1] Lautieri S De-risking Safety *IET Computing and Control Engineering June/Jul 2006 p38-41*
- [2] Lautieri S, Cooper D, Jackson D 2005 SafSec: Commonalities Between Safety and Security Assurance in *Proceeding of 13th Safety Critical Systems Symposium Southampton February 2005 Springer-Verlag*
- [3] Ainsworth A, Simpson A Integrated Modular Avionics — A View on Safe Partitioning, in *Towards System Safety, ed. Redmill and Anderson, Springer-Verlag, 1999.*
- [4] Cockram T, Is this the right room for an argument-Improving arguments for safety and security in *Safety and Reliability for Managing Risk, ed. Guedes Soares, Zio, September 2006 Taylor and Francis.*
- [5] <http://www.praxis-his.com/safsec/safSecResources.asp>
- [6] Froome P, Independent Safety Assessment of Safety Arguments in *Proceeding of 13th Safety Critical Systems Symposium Southampton February 2005 Springer-Verlag*
- [7] YSE. *Goal Structured Notation Handbook*, York Software Engineering Ltd., 1997
- [8] Fenn J et al. Safety Case Composition Using Contracts - Refinements based on Feedback from an Industrial Case Study in *Proceedings of 15th Safety Critical Systems Symposium (SSS'07), February 2007 Springer*