Combined security and schedulability analysis for MILS real-time critical architectures

Ill-ham Atchadam
University of Brest, Lab-STICC, CNRS UMR 6285, France
ill-ham.atchadam@univ-brest.fr

Frank Singhoff
University of Brest, Lab-STICC, CNRS UMR 6285, France
frank.singhoff@univ-brest.fr

Hai Nam Tran
University of Brest, Lab-STICC, CNRS UMR 6285, France
hai-nam.tran@univ-brest.fr

Noura Bouzid
University of Brest, Lab-STICC, CNRS UMR 6285, France
noura.bouzid@univ-brest.fr

Laurent Lemarchand
University of Brest, Lab-STICC, CNRS UMR 6285, France
laurent.lemarchand@univ-brest.fr

Abstract

Real-time critical systems have to comply with stringent timing constraints, otherwise, disastrous consequences can occur at runtime. A large effort has been made to propose models and tools to verify timing constraints by schedulability analysis at the early stages of system designs. Fewer efforts have been made on verifying the security properties in these systems despite the fact that sinister consequences can also happen if these properties are compromised. In this article, we investigate how to jointly verify security and timing constraints. We show how to model a security architecture (MILS) and how to verify both timing constraints and security properties. Schedulability is investigated by the mean of scheduling analysis methods implemented into the Cheddar scheduling analyzer. Experiments are conducted to show the impact that improving security has on the schedulability analysis.

2012 ACM Subject Classification Software and its engineering; Computer systems organization → Real-time system architecture

Keywords and phrases MILS (Multi Independent Levels of Security), RTCS (Real-Time Critical Systems), Security architecture and models, Scheduling analysis, Security analysis

Digital Object Identifier 10.4230/OASIcs.CERTS.2019.1

Acknowledgements This work and Cheddar[1] are supported by Brest Métropole, Ellidiss Technologies, CR de Bretagne, CD du Finistère and Campus France PESSOA programs 27380SA and 37932TF.

1 Introduction

Real-time critical systems (RTCS) are systems characterized by the temporal behaviors of their functions. The functions must imperatively respect deadlines specified by designers and serious disasters may occur if these deadlines cannot be met at runtime.

Although the respect of deadlines is an important issue in RTCS design, the security has also to be considered. The general purpose of securing a system is to prevent information

[1] `http://beru.univ-brest.fr/~singhoff/cheddar/`

disclosure (i.e. confidentiality) and preclude information's alteration (i.e. integrity) [22].

Security architecture models such as MILS (Multiple Independent Levels of Security) have been designed to make RTCS compliant with security objectives [12]. However, it is a challenge to design RTCS architectures with both resources management to enforce deadlines and also security policies to ensure security properties.

Schedulability can be enforced by a proper assignment of the tasks on the processors [13]. However, task assignment may lead to modify the RTCS architecture because of the assignment of security levels to the components. Indeed, assignment of security levels may require extra components to avoid violation of the information flow security. Then security architecture decisions may impact schedulability of RTCS and some means have to be proposed to RTCS designers to predict the impact of security-driven architectural design choices on the task schedulability, i.e. on the met deadlines of RTCS.

The ultimate goal of our work is to provide methods and tools for the exploration of real-time critical and secure architectures. Starting from a predefined model with real-time and security specifications, we want to automate the search of the design space defined by the constraints embedded in those specifications. It allows us to propose a set of solutions that represent trade-offs between security and schedulability requirements. For the first steps, in this article, we propose to evaluate the conflict aspects between security and schedulability concerns.

In this article, we propose to integrate a security architecture (MILS) in the Cheddar schedulability analyzer, a tool allowing RTCS designers to predict if the deadlines will be met at runtime. We extend the modeling capabilities of Cheddar [21] to model MILS security aspects and we ensure RTCS security properties verification by implementing several security models (Bell-La Padula, Biba). The resulting tool allows RTCS designers to run combined security and schedulability analysis. In addition, in order to evaluate the impact that improving systems security has on RTCS schedulability, we build a complete case-study used to conduct some experiments. Finally, with these experiments, we show the drawback of enforcing security properties for the RTCS schedulability.

The rest of the article is organized as follows. Section 2 gives an overview of security architectures, security models and the MILS architecture. Section 3 presents the Cheddar scheduling analyzer and depicts the system model and assumptions considered in our work. In addition, we propose our extension of Cheddar to model the MILS architecture. Section 4 details our approach of implementing security models which help to verify MILS architecture compliance and to compute number of security violations. We present a case study and our experiments to validate our contributions aforementioned in Section 5. Finally, Section 6 discusses related work and Section 7 concludes the article.

2 Background

This section introduces the main aspects relevant to security architecture and models. An overview of a high-assurance security architecture named MILS is also presented.

2.1 Security architecture and models

Regarding architecture designing, the concept of layering consists on separating hardware and software features into modular levels (Hardware, Kernel and Device Drivers, Operating System and Applications) [6]. Systems may use classifications, such as United States government classification system [16], which is based on the degree of secrecy and level of sensitivity. Classification levels can be confidential, Top_secret and Secret. They can be applied to

subjects or objects. Objects can be data classified at one level and subjects apply operations such as read, write or execute on objects.

A security model [22] describes the security strategy for a system with the purpose of ensuring security objectives (confidentiality, integrity). It is an implementation of some mathematical and analytical assumptions mapped to a system specification for resolving security issues. Examples of security models are Graham-Denning model [10], information flow control (IFC) models [22], State-Machine model [14], non-Interference model [9]. Bell-La Padula [3] and Biba [4] are concrete examples of IFC models.

A security architecture [6] uses an integrated view of the system to comply with security requirements. MLS (Multiple Levels of Security) and MILS (Multiple Independent Levels of Security) are examples of such security architecture [1].

2.2 MILS architecture

MILS is a high-assurance security architecture characterized by untrusted and trusted components, based on security models such as IFC [12]. It ensures confidentiality [22] which is the insurance of preventing the system from information disclosure and integrity [4] which is the protection of the system from unauthorized alteration. To meet those requirements, MILS is based on a set of properties named NEAT (Non-by-passable, Evaluatable, Always-invoked, Tamperproof) [2].

In order to ensure IFC, MILS adopts a classification level for subjects and objects which is Top_secret, Secret, Confidential and Unclassified for confidentiality; and High, Low, Medium for integrity. MILS is also based on the divide and conquer approach in order to reduce the effort for a system's security evaluation.

MILS introduces the following concepts [12]:

- Processes, which model the applications in a RTCS. For IFC purpose, they have to be labeled by a confidentiality level (e.g. Top_secret) and an integrity level (e.g. High). The same confidentiality and integrity levels are also applied to objects.
- Partitions, which are units of separations that host processes and/or data. They are characterized by a resource allocation in the space and time domains. Communications between processes in a partition and between partitions are subjected to IFC.
- Middleware Service Layer, which is a service responsible for maintaining the IFC. To enforce IFC, such component applies restrictions and permissions expressed by the designer according to communication between messages and processes and between processes themselves.
- Objects (*message*, *buffer*), communications.
- Application layer, which concerns all the processes. The components of this layer can be of 3 types[1]: Single Level of Security (SLS), Multiple Levels of Security (MLS), or Multiple Single Level of Security (MSLS).

3 MILS architecture modeling in Cheddar

In this section, we give explanations and details about Cheddar, a real-time scheduling analysis tool. We describe next how our models are represented and some assumptions adopted for this purpose. Last, we present the MILS modelling in Cheddar.

3.1 Cheddar scheduling analyzer

Cheddar [21] is a free real-time scheduling analysis tool. It provides a graphical editor and a library of schedulability analysis modules. The entry point of the tool is an architecture model expressed with Cheddar ADL. Cheddar ADL is a dedicated language designed to model software architecture of RTCS for scheduling analysis with Cheddar.

Cheddar ADL's basic entities can be grouped into 2 types: hardware and software components. The formers model the execution platform of the RTCS to be analyzed. Such components allow the designer to model processors, cache units, cores or any computing units, memory units and communication units. Software components model the software entities, i.e. applications composing a RTCS. Those components can be entities to model flows of control (Cheddar ADL *Task* entity), shared resources (entity *Resource* or *Buffer*) or task dependencies (Cheddar ADL entity *Dependency* or *Message*).

From a Cheddar ADL model, Cheddar provides two kinds of analysis tools: feasibility analysis tool, which assesses schedulability by analytical methods; and simulation analysis tool, which assesses schedulability by scheduling simulations.

3.2 System model and assumptions

We assume a RTCS consisting of tasks scheduled by a preemptive fixed priority scheduler. The system is compliant with the MILS architecture and is defined as follows:

- $\tau_1, \tau_2, ..., \tau_i, ..., \tau_n$ are the n tasks composing the system. Each task models a MILS process.
- T_i, C_i, D_i are respectively the period, the capacity (or the worst-case execution time) and the deadline of task τ_i. Any task τ_i makes its initial request at time 0, and then released periodically every T_i units of time. The task set is then synchronous. A task requires C_i units of computation time and must complete before D_i units of time.
- O_1, O_2, ..., O_i, ..., O_m are the m MILS objects in the system. Objects model messages exchanged between the tasks. Tasks can have read or write access to messages.
- IL_i and CL_i are respectively τ_i or O_i integrity and confidentiality levels.
- $\text{dep}(\tau_i, \tau_j)$ describes a communication from task τ_i to task τ_j.
- $\text{dep}(\tau_i, O_j)$ describes a communication from task τ_i to object O_j, task τ_i having write access to the object O_j.
- $\text{dep}(O_j, \tau_i)$ describes a communication from object O_j to task τ_i, task τ_i having read access to the object O_j.

In order to model MILS architectures with Cheddar, we need to make some extensions to Cheddar ADL, even if several Cheddar's components can be already used to model few MILS's entities:

- MILS partitions can be modeled by Cheddar *Address_space* entities.
- MILS objects can be modeled by Cheddar shared resources (i.e. *Buffer* or *Message* entities).
- MILS processes can be represented by Cheddar $Tasks$.
- MILS communications have the same semantics than Cheddar ADL dependencies. MILS messages can be modeled by asynchronous communication Cheddar dependencies.
- MILS functions do not need to be represented in Cheddar as they can be modeled by groups of partitions (i.e. groups of Cheddar Address Spaces).
- Finally, applications (which can be composed of one or more processes) are modeled by Cheddar ADL *Task* entities.

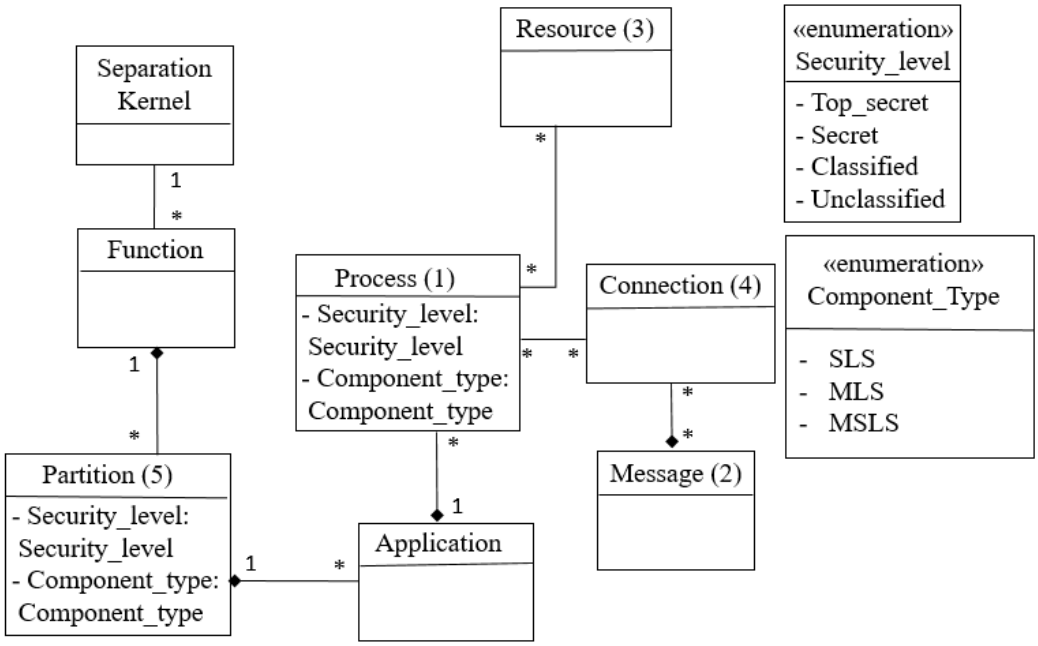

Figure 1 Modeling of MILS

To complete the modeling capabilities of Cheddar for MILS (Fig. 1) architectures, several Cheddar ADL entities have to be extended with new attributes modeling MILS data. As an example, we need to model *Buffers* and *Messages* confidentiality and integrity levels and also the right levels of tasks and partitions that are using them, i.e. if a partition or a task is allowed to handle a *Buffer* or a *Message* according to its authorization levels. We give bellow the list of properties we actually added to Cheddar ADL entities:

- An attribute named *Confidentiality_Level* (Top_secret, Secret, Classified, Unclassified) has to be defined for each Cheddar ADL address spaces, objects and tasks entities.
- An attribute named *Integrity_Level* (High, Medium, Low) also has to be defined for each Cheddar ADL address spaces, objects and tasks entities.
- *MILS_component_type* (SLS, MLS, MSLS) is an attribute to model MILS type of security level. Again, such attribute has to be defined in tasks and address spaces Cheddar ADL entities.
- Finally, *MILS_compliant_type* (Non_Compliant, Partition,...) specifies if a Cheddar's entity models a MILS's component or not. Such attribute is defined in any Cheddar ADL entity.

4 MILS security model implementation

In order to verify MILS architectures, we have implemented the two well-known security models Bell-La Padula [3] and Biba [4] into Cheddar. Both of them have been adapted and implemented in order to verify Cheddar ADL models. In the sequel, we describe their implementation as functions that take as entry a Cheddar model composed of cores, processors, address spaces, buffers, tasks, and dependencies.

4.1 Bell-La Padula in Cheddar

This security model was introduced to formalize the U.S. Department of Defense (DoD) multilevel security [16]. It is based on the *No read up, No write down* principle. No read up refers to the fact that a subject at a given security level cannot read data that is tagged with a higher security level. No write down means that a subject tagged with a given security level cannot write information to a lower security level.

We have implemented the Bell-La Padula algorithm in Cheddar. It checks if a Cheddar ADL model conforms to Bell-La Padula rules by returning the number of No read-up/No write-down rule violations. The algorithm is sketched below:

```
Sys: a Cheddar model composed of n tasks (τ1, τ2, ..., τi, ..., τj, ..., τn)
num: number of rules violations
For each dep in Sys loop
        If ((dep = dep(Oj, τi)) OR (dep = dep(τi, Oj)))
        AND (CLi < CLj) Then
                num:= num+1
        Elsif (dep = dep(τi, Oj)) AND (CLj < CLi) Then
                num:= num+1
        Elsif (dep = dep(τi, τj)) AND (CLj < CLi) Then
                num:= num+1
        End if
End loop
Return (num)
```

4.2 Biba in Cheddar

This security model was developed to ensure data integrity [4]. It is based on the *No read down, no write up* principle. "No read down" means that a subject cannot read data from a lower integrity level and "No write up" means that a subject cannot write data to an object tagged with a higher integrity level. We have implemented Biba rule checking into Cheddar. The algorithm is sketched below as a function that checks if a Cheddar model conforms to Biba rules by returning how many times the Read down and Write up rules are missed.

```
Sys: a Cheddar model composed of n tasks (τ1, τ2, ..., τi, ..., τj, ..., τn)
num: number of rule violations
For each dep in Sys loop
        If (dep = dep(Oj, τi)) AND (ILj < ILi) Then
                num:= num+1
        Elsif(dep = dep(τi, Oj)) AND (ILi < ILj) Then
                num:= num+1
        Elsif (dep = dep(τi, τj)) AND (ILi < ILj) Then
                num:= num+1
        End if
End loop
Return (num)
```

4.3 MILS securing process

In order to design MILS architectures that meet the different security rules previously presented, we propose to add the following components to the architecture model.

When a communication between two tasks of different security levels doesn't meet a security rule, we propose to add an encrypter or a decrypter components between the two tasks. So the tasks can still communicate but the data sent are encrypted and the designer can decide to allow communications by using a decrypter on the other side of the communication link. In MILS, such an encrypter (resp. a decrypter) is called a downgrader (resp. an upgrader).

In our implementation, downgraders and upgraders are extra Cheddar ADL tasks. Each of them has a period, a capacity, a deadline, a confidentiality and an integrity level. Assuming a Cheddar model composed of a set of tasks and their related dependencies, we run through these dependencies and check if they meet Bell-La Padula or Biba rules. If not, we add a downgrader or upgrader depending if it is Bell-La Padula or a Biba violation.

For such purpose, we make an arbitrary assumption about the downgrader (resp. upgrader) parameters: downgraders (resp. upgraders) inherit from receiver task period and deadline.

Furthermore and for the sequel experiments, we have chosen for encryption the AES/GCM (2K tables) algorithm (cycle per Byte: 16.1; cycles to set up key and IV(Initialization Vector): 3227) [7]. By supposing that our data size is 64 Bytes and that the frequency is 125 MHz, then the encrypter's execution time is 1660 us.

5 Experiments & Evaluation

The objective of our experiments is to evaluate the integration of the security architecture MILS into Cheddar and more precisely the two security models Bell-La Padula and Biba implemented into Cheddar. We also evaluate the impact of improving RTCS security by applying MILS can have on the RTCS schedulability.

We assume a single processor execution platform, on which all the tasks are placed in the same partition. The task set is synchronous and scheduled by preemptive fixed priority scheduler. Each task has two possibilities of *Confidentiality_Level* (Top_secret or Secret) and *Integrity_Level* (High or Medium).

The experimentation process is built by the three following parts: non secure models generation, secure models generation and models evaluation.

In the remainder of this section, we present in 5.1 a case study made for our experiments purpose. Sections 5.2 and 5.3 refer to the models generation part of the experiments. Finally section 5.4 gives the evaluation of the generated models and also the results of the experiments.

5.1 Case study

To conduct the experiments, we have created the study case 2 inspired by a drone system. It is made of two applications composed of dependent tasks: (1) a digital signal processing application called Constant False Alarm Rate detection (CFAR) [19]. It is a set of low critical tasks designed to detect target based on the variation of background noise. (2) A flight control system called ROSACE (Research Open-Source Avionics and Control Engineering)[17]. It is a longitudinal and multi-periodic flight controller designed as a benchmark to respond to stability, real-time and performance issues. ROSACE is composed of high critical tasks.

The set of tasks and their parameters of our case study are described in Table 1. This table does not give confidentiality and integrity levels of tasks since they are decided later in our experiments.

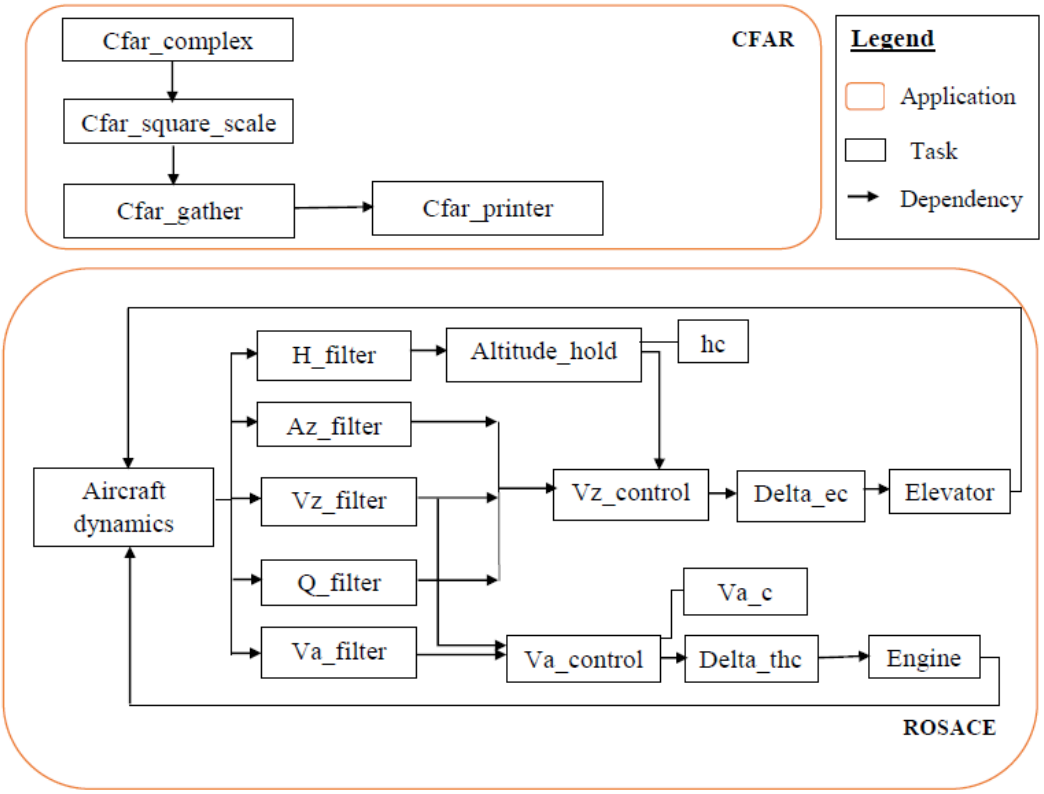

Figure 2 Case study model

Tasks	Period/deadline [us]	Capacity[us]
Cfar_complex	10000	90
Cfar_square_scale	10000	50
Cfar_gather	10000	340
Cfar_printer	10000	30
Aircraft dynamics	5000	200
Va_c, h_c	20000	500
H_filter, Az_filter, Va_filter, q_filter, Va_filter	10000	100
delta_e_c, delta_th_c	20000	500
Altitude_hold, Va_control, Vz_control	20000	100
Engine, Elevator	5000	100

Table 1 Case study parameters

5.2 Non secure models generation

This part consists of generating as many models as possible without considering if the generated models are secure or not. Our starting point is to describe a system. We used the case-study described in Section 5.1 with its default parameters of Table 1.

Confidentiality level of each task can be selected between two levels (Top_secret and Secret). The same holds for the integrity level (High and Medium). So our generator, for Bell-La Padula (resp. Biba) evaluation, takes the case-study model with an integrity (resp. confidentiality) level fixed for all the tasks and switches the confidentiality (resp. integrity) level of task. Each switching leads us to another different model. With 19 tasks in our case-study, we produce 1048576 ($2*2^{19}$) models for both Bell-La Padula and Biba evaluation.

5.3 Secure models generation

It consists of applying MILS securing process 4.3 on each non secure model generated in the previous step. To make all the non secure models secure, whenever a dependency doesn't respect Bell-La Padula or Biba's rules, we add a downgrader or an upgrader as additional task. For a given model, the number of downgraders or upgraders added is equal to the number of security rules violations.

As described in Section 4.3, we choose for encryption an AES/GCM (2K tables) algorithm with an execution time of 1660 us. Furthermore, we make an arbitrary assumption about the downgrader (resp. upgrader) attributes: downgraders (resp. upgraders) inherit from receiver task period and deadline.

5.4 Models evaluation

This section describes the evaluation of the schedulability and the security of all the generated models (non secure and secure). For each model, the security is quantified by the number of security violations while we decide to quantify the schedulability by the number of tasks missed deadlines.

It is important to notice that the number of security violations for secured models is null, while the number of missed deadlines will be null for the non secure models if the starting model is schedulable. Missed deadlines occur when adding security-related tasks (i.e. downgraders, upgraders).

As it can be noticed throughout this article, we have settled the security aspect into two parts: confidentiality and integrity. The figure (Fig. 3) below shows the relationship between scheduling (missed deadlines) and confidentiality (number of Bell-La Padula's rules violations).

In Fig. 3, each point corresponds to the number of confidentiality rules violations for a testcase (combination of different security levels) and the number of deadlines missed when we apply security rules to make the testcase secure by adding some downgraders.

The same work has been done for the integrity aspect. Figure 4) shows the relationship between schedulability (missed deadlines) and integrity (number of Biba's rules violations).

We observe that two different RTCS models with the same number of rules violations may not lead to the same number of missed deadlines. It is explained by the fact that downgraders/upgraders added to solve security problems may not have the same period and deadline in both RTCS models. Indeed, a downgrader's period and deadline depend on the period and the deadline of the sink task of the non-secure dependency.

From our experiments, with the security metrics, we conclude that the higher number of violations (confidentiality or integrity) is, the more we have tasks missing their deadlines

Figure 3 Schedulability and confidentiality evaluation

Figure 4 Schedulability and integrity evaluation

when adding downgraders/upgraders through the MILS securing process. Fig. 3 and 4 show an quantification of such impact.

Finally, we can observe in the figures that even if the Bell-La Padula and Biba results of our experiments look similar, for a RTCS that violates both Bell-La Padula and Biba rules, resolving the Bell-La Padula's problem does not solve automatically the Biba's one and conversely.

6 Related work

In [15], the authors worked on designing a security gateway in avionic domain based on Integrated Modular Avionics (IMA) architecture, characterized by the separation of system resources into partitions. In order to control information flow between partitions for security issues, they applied MILS principles based on the real-time OS named PikeOS.

In [8], the authors modeled MILS using AADL (Architecture Analysis and Design Language) [20] and proposed an automated implementation process based on code generation. Applications are run on a MILS operating system called POK. In [5], the authors presented a guide on securing a system design using MILS architecture. They focused their work on distributed systems and systems of systems.

The article [11] discusses how to build a secure system by using AADL. The authors proposed an approach to validate the confidentiality of systems by defining assumptions based on Bell-La Padulla's security protocol. They showed how security can have impact on data quality, resource consumption, availability, reliability, and real-time performances.

One important concern about MILS is task partitioning. [18] presents XtratuM, a hypervisor that helps to build partitioned systems while meeting safety critical real-time requirements. Scheduling analysis is performed using Xcronte. The authors also measure the cost in terms of performance and memory footprint of using XtratuM compared to other partition development environments.

Our approach is different of the above ones because we integrate MILS concepts in scheduling analysis as none of them have experimented joined schedulabity and MILS security analysis . As far as we know, it is the first time one extends a scheduling analyzer to support a secure architecture. We implemented into Cheddar several security models in order to jointly enforce MILS properties and schedulability.

7 Conclusion

In this article, we investigated security properties in RTCS through a high assurance security architecture MILS. Our work overlays real-time scheduling analysis, and architecture design and security.

We have proposed an extension of Cheddar to model MILS real-time critical architecture and to perform both schedulability analysis and security analysis. We have integrated MILS concepts including the two security models (Bell-La Padula and Biba) with Cheddar ADL.

We have evaluated the impact of security on schedulability analysis. From the experiments, we observed that securing a system can affect negatively its scheduling by leading some tasks missing their deadlines. We also have shown that downgraders and upgraders periods and deadlines have impact on numbers of missed deadlines. Finally, we observed that confidentiality and integrity are independent, meaning that resolving one does not help to resolve the other.

As future work, we want to measure more precisely the cost of securing a RTCS and explore trade-offs to optimize both security and scheduling with different encryption algorithms for downgrader and upgrader components.

References

1 Jim Alves-Foss, Paul W Oman, Carol Taylor, and Scott Harrison. The mils architecture for high-assurance embedded systems. *IJES*, 2(3/4):239–247, 2006.

2 R William Beckwith, W Mark Vanfleet, and Lee MacLaren. High assurance security/safety for deeply embedded, real-time systems. In *Proceedings of the Embedded Systems Conference*. Citeseer, 2004.

3 D Elliott Bell and Leonard J La Padula. Secure computer system: Unified exposition and multics interpretation. Technical report, MITRE CORP BEDFORD MA, 1976.

4 Kenneth J Biba. Integrity considerations for secure computer systems. Technical report, MITRE CORP BEDFORD MA, 1977.

5 Carolyn Boettcher, Rance DeLong, John Rushby, and Wilmar Sifre. The mils component integration approach to secure information sharing. In *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*, pages 1–C. IEEE, 2008.

6 Eric Conrad, Seth Misenar, and Joshua Feldman. *Eleventh Hour CISSP*. Elsevier, 2013.

7 Wei Dai. Crypto++ 5.6.0 benchmarks. URL: `https://www.cryptopp.com/benchmarks.html`.

8 Julien Delange, Laurent Pautet, and Fabrice Kordon. Design, verification and implementation of mils systems. In *Proceedings of the 21th International Symposium on Rapid System Prototyping*, pages 1–8, 2010.

9 Lei Gong, Lu Tian, and Fulian Zhang. Application information flow non-interference transmission model. In *Proceedings of 2011 Int. Conf. on Electronic & Mechanical Engineering and Information Technology*, volume 5, pages 2306–2309. IEEE, 2011.

10 G Scott Graham and Peter J Denning. Protection: principles and practice. In *Proceedings of Spring Joint Computer conference*, pages 417–429. ACM, 1972.

11 Jörgen Hansson, Peter H Feiler, and John Morley. Building secure systems using model-based engineering and architectural models. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2008.

12 OH Holger Blasum and S Tverdyshev. Euro-mils: Secure european virtualisation for trustworthy applications in critical domains–formal methods used.

13 Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

14 Francois Mouton, Alastair Nottingham, Louise Leenen, and HS Venter. Underlying finite state machine for the social engineering attack detection model. In *2017 Information Security for South Africa (ISSA)*, pages 98–105. IEEE, 2017.

15 Kevin Müller, Michael Paulitsch, Sergey Tverdyshev, and Holger Blasum. Mils-related information flow control in the avionic domain: A view on security-enhancing software architectures. In *IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops (DSN 2012)*. IEEE, 2012.

16 Barack Obama. Executive order 13526: Classified national security information. In *United States. Office of the Federal Register*. United States. Office of the Federal Register, 2009.

17 Claire Pagetti, David Saussié, Romain Gratia, Eric Noulard, and Pierre Siron. The rosace case study: From simulink specification to multi/many-core execution. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014 IEEE 20th*, pages 309–318. IEEE, 2014.

18 Ismael Ripoll, Miguel Masmano, Vicent Brocal, Salvador Peiró, Patricia Balbastre, Alfons Crespo, Paul Arberet, and Jean-Jacques Metge. Configuration and scheduling tools for tsp systems based on xtratum. *Data Systems In Aerospace (DASIA 2010)*, 2010.

19 Benjamin Rouxel and Isabelle Puaut. Str2rts: Refactored streamit benchmarks into statically analyzable parallel benchmarks for wcet estimation & real-time scheduling. In *17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

20 SAE. Architecture analysis & design laguage v2.0 (as5506), September 2008.

21 Frank Singhoff, Jérôme Legrand, Laurent Nana, and Lionel Marcé. Cheddar: a flexible real time scheduling framework. In *ACM SIGAda Ada Letters*, volume 24. ACM, 2004.

22 Ming-Xin Yang, Li-Na Yuan, and Zhi-Xia Yang. A discuss of computer security strategy models. In *2010 Int. Conf. on Machine Learning and Cybernetics*, volume 2, pages 839–842. IEEE, 2010.