A safe dynamic access control providing mandatory automotive cybersecurity

Vincent Hugot, Adrien Jousse, Christian Toinard, Benjamin Venelle

A safe dynamic access control providing mandatory automotive cybersecurity

Vincent Hugot
LIFO
INSA Centre Val de Loire
Bourges, France
vincent.hugot@insa-cvl.fr

Adrien Jousse*
GEEDS
Valeo
Créteil, France
adrien.jousse@valeo.com

Christian Toinard
LIFO
INSA Centre Val de Loire
Bourges, France
christian.toinard@insa-cvl.fr

Benjamin Venelle
GEEDS
Valeo
Créteil, France
benjamin.venelle@valeo.com

***Abstract*—The recent computerization of automotive vehicles offers new ways for attackers to penetrate critical systems. To ensure their safety, cybersecurity is therefore required. Cybersecurity can be enforced with different mechanisms, such as access control, thanks to reference monitors. However, systematic access control may harm safety properties in certain scenarios. Therefore, the integration of access control mechanisms in automotive systems remains an open problem. Our paper provides a general approach adding dynamic mandatory access control to enforce security properties while guaranteeing their innocuousness with respect to the vehicle's safety. To achieve this, we start by formally modeling the system. Then, a mandatory access control policy is proposed with respect to the different vehicle's states. Using a dedicated model checking tool, the safety of the access control mechanism is verified with respect to a formal attack model. Using the attack paths produced by the verification tool, the access control policy can be iteratively refined, thus improving the availability and resilience of the system.**

***Index Terms*—cybersecurity, automotive, reference monitor, model checking, modelization**

INTRODUCTION

In this paper, we propose to integrate reference monitors [1] into automotive systems to enforce cybersecurity while guaranteeing safety. The safety of the integration is assessed using model checking. Figure 1 summarizes our approach.

Today's vehicles offer a large attack surface to intruders. By exploiting it, they cross cybersecurity boundaries [2], interfere with offered functionalities, and put at risk the safety of the vehicle [3]. In [4], hackers successfully drove a car remotely by sending commands to the vehicle's Telematics and Communication Unit (TCU). In this scenario, every command sent to the TCU was executed, without any control. A reference monitor embedded in the TCU would have blocked the attack by enforcing specific cybersecurity properties (*e.g.* block irrelevant commands received by the TCU), thus preserving the integrity of the system. As every vehicle is approved [5], its safety should be guaranteed in any situation, even an attack, making cybersecurity mandatory. However, why would one trust that the addition of the reference monitor will actually enforce the desired property? Furthermore, what guarantees do we have that it does not interfere with the system's safety requirements or other functionalities? Our paper addresses those questions.

Reference monitors are poorly used in the automotive industry. For instance, the AUTOSAR consortium [6] does not specify implementation standards for reference monitors in automotive systems. Indeed, a reference monitor can interfere with safety and functional aspects, leading to undesirable effects. Previous works [7] on non safety-related systems have shown that systematic access control can render the system inoperative. Historical access control mechanisms [8] unconditionally authorize or deny actions based on a static policy. Hence, some actions may be denied because the control does not take the states of the system into account, leading to denial of service. With such mandatory access control, the static policy must authorize all the actions the system may need to perform, in every possible situation, loosening the policy. In contrast, our reference monitors use a state machine approach, where the authorised actions depend upon the current state. The dynamic nature of this type of access control makes it less permissive but harder to specify.

To avoid unwanted interference with safety properties, which a faulty dynamic access control might cause, we propose to formally verify the cybersecurity mechanisms with respect to the vehicle's safety properties, and against a given attack scenario. If the model checking tool detects violations of the requested integrity properties, the access control policy is inadequate with respect to the attack scenario. In that case, the policy must be improved accordingly. Our solution enumerates the threats, thus guiding the evolution of the mandatory control.

This paper describes our model, built with a tool developed by V. Hugot [9]. This tool uses Computational Tree Logic (CTL) formulæ. Compared to SPIN [10], [11], it has the advantage of featuring a graphical representation of the remaining threats, and is thus more user-friendly for industrial usage.

Our approach is a mixture of offline [12], [13] and online [14] verification. The main thrust is the online component: the reference monitor, which reacts to runtime events, may raise warnings, or even block certain messages depending on the vehicle state. The role of the offline phase is to make sure that the mandatory control does not interfere with the safety

Short paper submitted to CSCI-ISCW

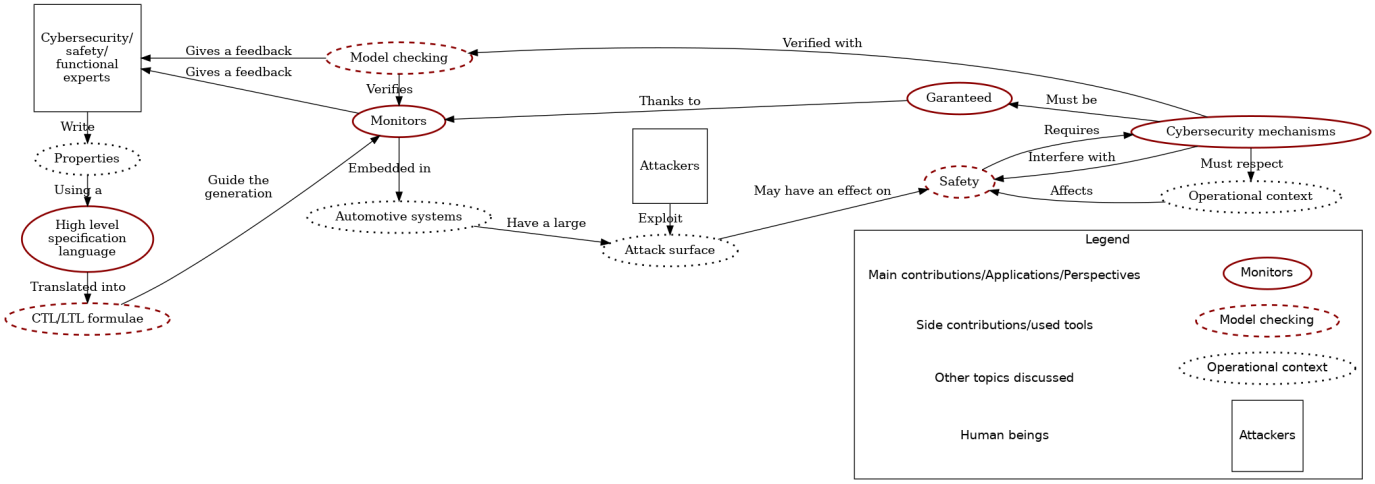

Fig. 1. Graph summarizing our approach

of the system — the functional properties are preserved. The offline phase , which provides a user-friendly view of available attack paths, is a valuable tool for improving the monitor. Several iterations can be necessary to optimize the design. This is, however, much more practical than testing the access control in a prototype or a simulator, which is more common practice in the automotive industry.

In practice the diversity of actors and suppliers makes a full formal specification difficult. This is not as much an obstacle for our mixed approach as it would be for pure offline verification: we need only focus on the aspects of the system which the monitor might interfere with.

We will illustrate our approach with an automotive example: an intelligent steering wheel (ISW). This steering wheel enables the driver to turn on an automatic driving mode. An indicator light informs the driver of the status of the automatic driving mode. As the light communicates to the driver whether he must pay attention to the road, it is of critical importance to the driver's safety that it be accurate. Our reference monitor is designed to protect the system against a generic attacker trying to switch ON the light in any state of the system. We can prove that, under that attack scenario, a large range of attacks is safely prevented.

This paper is organized as follows. The next section offers reminders about model checking and presents our CTL-based solution, which provides useful visual representations of the system and the impact of attacks. We then conclude with a few perspectives for future works.

Cybersecurity for an intelligent steering wheel

In this section, we provide basic information about model checking, describe our model and show that the integration of our access control prevents attacks while preserving the safety of the intelligent system.

Model checking performs an (explicit or symbolic) state space exploration representing the evolution of a system, given a formal specification of the system's operational rules. By exploring this model, we can assess whether the model satisfies the properties written by the cybersecurity, safety, and functional experts. In practice, as said above, most actors have only a partial (and often informal) view of *some* components. Here, we focus on the internal communications of our Tier1-level example of the ISW. The ISW is modelled as a *synchronized product* of sub-components (here, Electronic Control Units, or ECU). The general idea is that the behavior (including communications) of each individual ECU is modelled separately (as finite-state automata for now, *cf.* subfigures *Button automaton* and *Attacker automaton* of Figure 2).

Some of those communications are marked as requiring synchronization between components, others not. From this, the global behavior of the system, covering all possible interactions, is then mechanically derived, yielding a potentially much larger automaton. Examples of this global automaton are shown as background of Figure 2. These automata were generated using a CTL tool [9].

We first model (what we know of) our example system. The expected behavior of the ISW is as follows: in the initial state, the automatic driving is disabled, the light is OFF and both buttons are released. When both buttons are pressed at the same time, the mainboard sends a message to the automatic driving ECU to enable the automatic driving mode. When automatic driving is enabled, the mainboard sends a message to switch ON the light, which notifies the driver that automatic driving is enabled. Then, when both buttons are pressed again, the mainboard switches OFF the light and then disables the automatic driving mode, returning to the initial state.

This preliminary modeling step has some intrinsic value, as it may reveal some basic design flaws, even in nominal behavior. If flaws are found, experts can refine their properties (if they are improperly written) or make changes to the system. In the second step, we model and add the attacker (as one or several new subcomponents), thus revealing successful attacks (if any). In our model, the attacker tries to switch ON the light whenever he can. Practically speaking, in any state of the

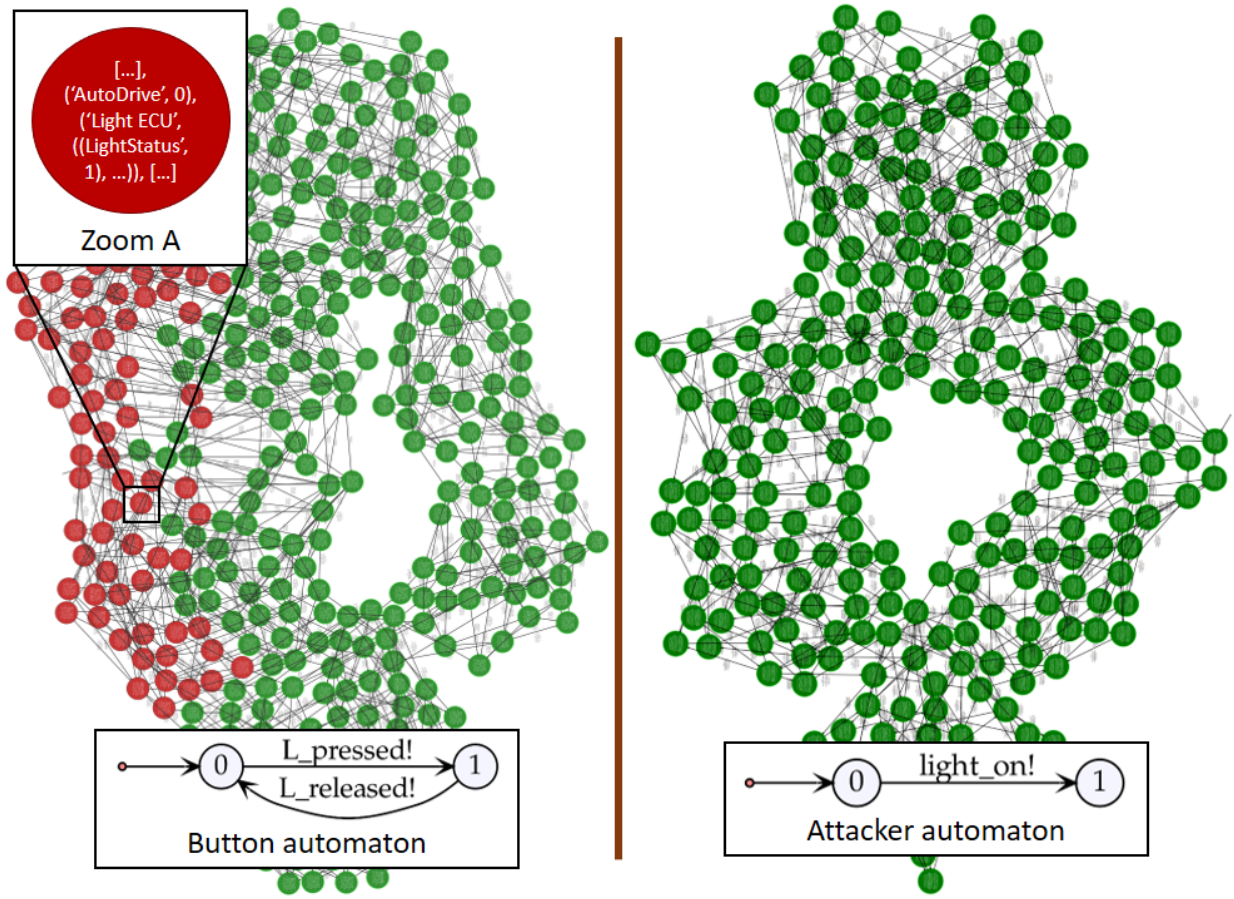

Fig. 2. Synchronized product automata of systems under attack (without our reference monitor on the left, with our reference monitor on the right) and examples of sub-system specifications on the bottom

global system, we consider that the attacker has the capability to ask for switching ON the light (*cf.* subfigure *Attacker automaton* of Figure 2). This covers a large range of attacks since we do not limit the way an attacker enters into the system. Attacker successes can be of two types:

1) he switches ON the light when the automatic driving mode is disabled (*i.e.* leading to an unsafe situation, outside of the nominal specification)
2) he switches ON the light when the automatic driving mode is enabled (*i.e.* leading to a safe situation, but it might bypass some transitions)

The left automaton of Figure 2 represents the global automaton of our ISW, under attack. This figure is the result of the synchronized product of all component specifications.

We evaluate the correctness of the model with respect to some properties written in temporal logic by cybersecurity, safety, and functional experts; for example, the safety CTL formula ϕ =(*Light* $\Rightarrow$ *AutoDrive*), which means "*Whenever the light is on, then automatic driving is enabled*". To ensure that our reference monitor does not interfere with the system's legitimate operations in the name of security, we also verify that the addition of the monitor preserves the system's functional properties such as: ψ =**EF**(*Light*), which roughly means "*There is always a way to turn on the light*".

The tool yields a colored representation (*cf.* Figure 2), indicating which states of the global system satisfy the properties ϕ, ψ (in green) or not (in red).

With an attacker and without the dynamic mandatory access control, we observe in Figure 2, Zoom A, a state where our property ϕ does not hold (*i.e.* the light is ON while the automatic driving is disabled). Thus, the attacker is successful (success type 1). This is a violation of a safety property due to a cybersecurity attack. ϕ is only violated when the light is ON and the automatic driving is disabled. All red states here have been made accessible by the attacker. (However, if the attacker switches ON the light when the automatic driving is already enabled, ϕ is still verified, success type 2.) By examining the different attack flaws, the designers can implement a dedicated dynamic policy (Figure 3).

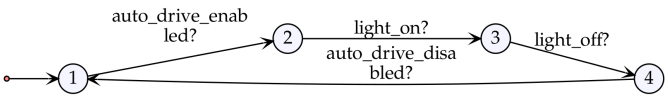

Fig. 3. Dynamic access control for the ISW

Thus, the designers adds a dynamic mandatory control

(Figure 3) that enforces the following sequence: automatic driving is enabled, light is switched ON, light is switched OFF, automatic driving is disabled.

The aim of our reference monitor is to preserve the integrity of the system regarding ϕ and ψ. Without the system's integrity with respect to ϕ, the driver may be fooled into thinking that the automatic driving feature is ON when it is not (as we saw previously, success type 1). Adding our reference monitor constrains the system's behavior: the mainboard, the light, and the monitor are synchronized. As we see on the right background of Figure 2, all states are green, meaning ϕ (and ψ) are always verified. The integrity of the system is preserved. The attack is foiled as the attacker cannot switch ON the light while the automatic driving is not enabled.

Moreover, we must ensure that the reference monitor does not block anything essential, meaning that ψ (among other safety and functional properties) must also be verified. If our reference monitor simply forbade turning ON the light, unconditionally, then the system would technically be secure in the sense of ϕ, but we would have lost a functionality, in the sense of ψ. To guarantee the innocuousness of the reference monitor, we check correctness of the monitored system (without the attacker) with respect to ψ and all desired safety and functional properties, thus ensuring that our reference monitor (*cf.* Figure 3) improves the cybersecurity of the vehicle without altering the safety of the system. As expected, ψ is always verified, meaning the system preserved its properties. If an incorrect access control policy was specified (*e.g.* not allowing the light to be switched ON) ψ would have been violated resulting as red states on our global system (*cf.* Figure 2).

That dynamic control can be automatically generated from a cybersecurity property expressed through a logic formula, which it enforces by construction, or it can be generated manually. While automatic generation is desirable, it is not always straightforward to put into practice. The target property may involve characteristics of the system that are not accessible by the ECU on which the monitor is installed. Thus, it must be rephrased, and possibly loosened, in terms of what can actually be observed. This aspect could be handled semi-automatically as well, as we hope to do in future works. However, the graphical view of the threats eases the designers analysis and several attempts of control can be sufficient to have a fully green check. Moreover, the designers can decide if the remaining threats have a strong probability of occurrence through exogenous methods of risk analysis.

Conclusion and future works

Model checking tools are not widely used or known in our industrial context. Yet, our experiments show that they can help improving cybersecurity of automotive vehicles while preserving their safety. In this proof of concept, we were able to: model the system, model some of its desired properties, and verify the required properties, both with and without our mandatory access control.

As shown, we can easily formalise an attack scenario covering a large range of threats, and design a reference monitor preserving both the integrity of the system and its safety under that scenario.

Larger systems may run afoul of the state-space explosion problem [15], and require the use of tools, such as SPIN [10] [11], that are more scalable, though potentially less user-friendly.

On that front, a possible approach is the design of a specification language to simplify the definition of the system's properties. Such a language could encompass safety, functional, and NIST cybersecurity properties [16]. It could mechanically translate them into CTL properties (or other suitable logics).

Our current model is fully synchronized, and thus does not fully reflect real-world behavior, which is a potential limitation. Our monitor needs to be able to handle any loss or desynchronization that might occur. We have started testing with outages and transient failures. With the addition of loss detection and/or replay, results are promising and must be confirmed on real-life automotive systems.

References

[1] J. P. Anderson, "Computer Security Technology Planning Study," Tech. Rep. ESD-TR-73-51, U.S. Air Force Electronic Systems Division, 10 1972.

[2] V. Hugot, A. Jousse, C. Toinard, and B. Venelle, "oMAC : Open Model for Automotive Cybersecurity," in *17th escar Europe : embedded security in cars (Konferenzveröffentlichung)*, (Stuttgart, Germany), pp. 170–184, Nov. 2019.

[3] Y. Shoukry *et al.*, "Non-invasive spoofing attacks for anti-lock braking systems," in *Cryptographic Hardware and Embedded Systems - CHES 2013*, (Berlin, Heidelberg), pp. 55–72, Springer Berlin Heidelberg, 2013.

[4] A. Greenberg, "Hackers remotely kill a jeep on the highway - with me in it." Online, July 2015.

[5] ISO, "26262-1:2011 Road vehicles – Functional safety," 2011.

[6] "Autosar." Online.

[7] B. Venelle, J. Briffaut, L. Clévy, and C. Toinard, "Security Enhanced Java: Mandatory Access Control for the Java Virtual Machine," in *ISORC - 6h IEEE International Symposium on Object, Component, and Service-Oriented Real-Time Distributed Computing - 2013*, (Paderborn, Germany), June 2013.

[8] NSA, "Selinux project." Online, December 2000.

[9] V. Hugot, "Nfa framework for 4a class on verification / model-checking." Online, October 2020.

[10] G. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 1st ed., 2011.

[11] G. Holzmann, "The model checker spin," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[12] E. A. Emerson and E. M. Clarke, "Characterizing correctness properties of parallel programs using fixpoints," in *Automata, Languages and Programming* (J. de Bakker and J. van Leeuwen, eds.), (Berlin, Heidelberg), pp. 169–181, Springer Berlin Heidelberg, 1980.

[13] J. P. Queille and J. Sifakis, "Specification and verification of concurrent systems in cesar," in *International Symposium on Programming* (M. Dezani-Ciancaglini and U. Montanari, eds.), (Berlin, Heidelberg), pp. 337–351, Springer Berlin Heidelberg, 1982.

[14] M. d'Amorim and G. Roşu, "Efficient monitoring of ω-languages," in *Computer Aided Verification* (K. Etessami and S. K. Rajamani, eds.), (Berlin, Heidelberg), pp. 364–378, Springer Berlin Heidelberg, 2005.

[15] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, *Progress on the State Explosion Problem in Model Checking*, pp. 176–194. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[16] NIST, "Glossary csrc." Online, June 2021.