Red-Zone: Towards an Intrusion Response Framework for Intra-Vehicle System

Mohammad Hamad, Marinos Tsantekidis and Vassilis Prevelakis
Institute of Computer and Network Engineering, TU Braunschweig, Germany
{*mhamad, tsantekidis, prevelakis*}*@ida.ing.tu-bs.de*

Keywords: Security, Automotive Security, Intrusion Response System, Insrution Detection System, Red-Zone

Abstract: Modern vehicles are increasingly equipped with highly automated control systems both for driving and for passenger comfort. An integral part of these systems are the communication channels that allow the on-board systems to interact with passenger devices (e.g. tablets), ITS systems (e.g. road-side units), and other vehicles. These advances have significantly enlarged the attack surface and we already have numerous instances of successful penetration of vehicular networks both from inside the vehicle and from the outside. Traditional mechanisms for detecting and responding to such attacks are ill-suited to the vehicular domain mainly due to the fact that the entire process of dealing with an attack must be handled automatically and in a way that does not affect safety or severely impacts the continued availability of the vehicle or its key systems. Once a security breach is suspected, the system must evaluate the circumstances in order to determine whether the threat is real (and not a false positive) and select the optimal response through the use of an Intrusion Response System (IRS). Although IRSs have been adopted in other domains, there is a lack of such systems in the vehicular field. In this paper, we investigate the challenges and requirements for integrating such a mechanism inside a vehicle. In addition, we present an Intrusion Response System based on the Red-Zone principle which meets the identified requirements. Finally, we discuss the integration of IRS through the vehicle system development and the different aspects which support such a process.

1 INTRODUCTION

Recently, vehicle manufacturing has changed significantly. These changes were clearly reflected by the increasing complexity of the modern vehicle system. Nowadays, vehicles are becoming like a network of connected computers on wheels. A contemporary vehicle contains more than 70 microcontroller-based computers known as Electronic Control Units (ECUs) which are distributed all over the vehicle, interconnected via different types of networks such as Controller Area Network (CAN), Flexray, and Ethernet (Broy, 2006). In addition, the software in each car comprises millions of lines of code (LOC) (Charette, 2009) which are integrated within the various ECUs to implement the different functions in the car, ranging from very simple ones such as controlling courtesy lights to highly critical ones such as engine control. Modern vehicles also rely on many external interfaces, such as bluetooth, WiFi, GPS, etc, to interact with the outside world and communicate with other vehicles, passenger devices and road side units to improve vehicle and road safety, traffic efficiency, as well as comfortability of both drivers and passengers.

However, the enormous growth of vehicle functionalities, capabilities and connectivity was accompanied by new security vulnerabilities which expanded the vehicle's attack surface and made it a very attractive target to adversaries (Koscher et al., 2010; Checkoway et al., 2011; Hamad et al., 2016a). Attackers were able to exploit these vulnerabilities mounting serious attacks without the need of physical access to the vehicle (Ishtiaq Roufa et al., 2011). Therefore, developing mechanisms to prevent and/or detect such attacks became a very crucial requirement (Larson and Nilsson, 2008). As a first line of defense, many prevention mechanisms were adopted to guarantee data confidentiality and integrity and to prevent unauthorized third parties from accessing services and communication channels within the vehicle. Since security can never be absolute, Intrusion Detection Systems (IDSs) were introduced as a second layer of defense. The IDS is used to monitor the vehicular system and its network to detect any violation of a predefined security policy or any malicious

behavior of the system components during operation (Hamad et al., 2016b; Larson et al., 2008). Whenever an attack is detected, the IDS has to alert the system in order to react accordingly.

The embedded nature as well as the safety-critical aspects of the vehicle make the response to the detected attack as critical as the detection of the attack itself. The typical system response when detecting an attack to a component is, optionally, to restart that component (Strasburg et al., 2009) in the hope that the failure, which was caused by the attack, was a transient one. For critical-safety systems, this default response is not satisfactory or even desired because it may cause an accident if a certain component stops working (Le Lann, 1996). The security policy of the vehicle has to implement different strategies in order to react to an attack, strategies which need to ensure high system resilience and safety. The response strategies are implemented through a so-called Intrusion Response System (IRS). Generally - and even more so in the vehicular domain - IRSs have received less attention and research effort compared to IDSs (Stakhanova et al., 2007) until now.

Contribution: In this paper, we discuss the main requirements and challenges that the adoption of an IRS within the vehicular domain faces. In addition, we propose an Intrusion Response Framework for intra-vehicle systems based on the Red-Zone principle. We discuss the different aspects that need to be considered during the development of such a framework.

The rest of the paper is organized as follows. Section 2 mentions the current state of affairs concerning IRSs. In Section 3, we present the challenges IRSs face in a vehicular context and the requirements of such a system. In Section 4, we present a general overview of the Red-Zone principle and the IRS based on that principle. The development of the IRS and the different related aspects are explained in Section 5. In Section 6, we discuss the development process as well as the proposed responses using autonomous driving as a use-case. Finally, we conclude our paper in Section 7.

2 INTRUSION RESPONSE SYSTEMS

The growth of hacker knowledge, expertise and tools put cyber systems in continuous danger, as indicated by the increasing incidents of attacks each year. Consequently, intrusion prevention mechanisms (firewalls, cryptography, access control, etc.) alone are not sufficient to mitigate these attacks. IDSs were widely developed to detect, analyze and report intrusions in a computing system. Whenever a task behaves abnormally or violates a predefined security policy, the IDS considers this task as a malicious one. Some IDSs have already implemented limited static responses, such as generating an alarm or report (Kemmerer and Vigna, 2002). However, with increasing levels of attacks' complexity and targeted domains, more comprehensive response strategies are required. These strategies could be implemented through Intrusion Response Systems (IRS).

Authors in (Stakhanova et al., 2007; Shameli-Sendi et al., 2012; Inayat et al., 2016) have surveyed the existing IRSs. They have proposed a taxonomy of these systems according to different characteristics:

Time of response: this aspect determines when the responding action should take place. IRSs either activate the response after the occurrence of the attack (delayed response), or take action before the attack has affected the system resources (proactive response).

Nature of response: Determines the activity of the selected response. IRSs can issue a response which aims to limit the effect of the attack (active response). Another response could only notify the system (passive response) without any further actions.

Degree of automation: This characteristic defines how the response is taking place. Some IRSs require the interference of the system administrator in order to apply the predefined response (manual response), while others are totally independent and do not require any human interaction to react to an intrusion (automatic response).

Response selection: Authors distinguish between the different IRSs based on the way these systems choose the applied responses. Most of the IRSs, such as (Locasto et al., 2005), are using a fixed predefined response for a certain alert (static response). A number of IRSs choose the response based on attack metrics, therefore the response for the same attack could be different from one instance to another (dynamic response).

3 IRS FOR VEHICLES

Most of the existing IRSs are mainly deployed for normal network systems or usual computer systems (e.g. (Sterne et al., 2001),(Herold, 2017), etc.). Within the vehicular domain, very few authors have investigated the design of an IRS. Even in these few proposals, authors have looked at the response as

a part of the intrusion detection framework (Hoppe et al., 2008). Recently, Vőlep et al. (Vöelp and Esteves-Verissimo, 2018) have proposed an intrusion-tolerant architecture to tolerate partial compromise of software components of autonomous vehicle. In other case, e.g. (Nadeem and Howarth, 2014), intrusion detection and adaptive response mechanism were designed to detect a range of attacks and to provide an effective response for Mobile Ad hoc Networks (MANETs). However, the vehicle, which is considered as a safety-critical system, has its own special properties and restrictions which limit the adoption of the existing IRS of the other domains. In the next subsection we explain some of these challenges which affect the design of any intrusion response framework for the intra-vehicle system.

3.1 Challenges

Highly interconnected architecture: As we mentioned in the intro, a vehicle contains a huge number of ECUs which are supplied by different vendors with widely varying degrees of security awareness. These ECUs need to collaborate with each other, therefore many of them are connected with various sub-networks and share the same bus system with each other. The unrestricted interaction among those ECUs puts the whole system in danger. Any attack on one of these ECUs could cause a domino effect of attacks on the whole system (Koscher et al., 2010). Consequently, any response or mitigation mechanism should consider the highly interconnected and distributed nature of a vehicular environment.

Ever-changing scenery: Cars are designed to work for an average of 12 years (Markit, 2016). So, a car in its lifespan may have updates in both its hardware and software components. This means that the rules of the security policies in an IRS cannot be static, but must be defined as dynamic and changing in order to accommodate the behavior of the newly introduced components as well as the new discovered threats.

Autonomous and semi-autonomous nature: Unlike in a traditional IT environment where a human administrator would be expected to approve and apply the response, in the vehicular environment this may not always be applicable in the case of an autonomous car. Nowadays, some might consider the driver as the administrator of the vehicular environment. This assumption is not preferable because the driver's attention must not be diverted while driving, for safety reasons. Moreover, she may not have the technical knowledge about the attack and the required response.

3.2 Requirements

The existing IRSs, which are deployed on the other domains, fulfill only a subset of the aforementioned challenges. Therefore, based on both the challenges of the vehicular domain and the general IRS taxonomies (see Sec. 2), we propose here the desired properties for any IRS which is developed for the intra-vehicle system. The proposed IRS should be:

Proactive: An IRS should be designed to predict an attack on the system and not wait until after the attack takes place to detect it. Early prediction provides the system with a sufficient amount of time to respond in an effective manner. However, complications arise from the possibility that we are responding to a false positive. Unfortunately, as the security constraints are tightened, the likelihood of false positives increases.

Active: The proposed system should react in a way that mitigates the damage caused by the attack and prevents its propagation to other subsystems. In addition, it must consider issues such as containment, continued availability, interaction with other subsystems and, in certain cases, latency. These requirements are in many cases in conflict with each other; for example, security and availability often work against one another) (Tryfonas et al., 2000).

Correct: The IRS should react properly, based on the type and target of a potential attack. In a case if the IRS detects that a component is acting off-nominally, it responds to that behavior by terminating (killing) the running process and executing a new instance of it. However, such response is not preferable because it may cause an accident. On the other hand, because of the lack of direct interaction with the driver to solve the security issue, terminating the malicious task without any relevant information about the vulnerability may not be the right choice, as an attacker could use the same obscure vulnerability to break the newly instantiated task repeatedly.

Heterogeneous: The IRS should be designed to respond in multiple ways, including a fully automated response. When a task is terminated, regardless of the cause of termination, security related or otherwise, the system should respond to the failure by, (i) using an alternate system to provide the functionality that is lost by the demise of the failed task, (ii) determining whether the system can continue running without this functionality (fail-continue), or (iii) forcing the system into a fail-safe condition, i.e. it goes into a mode where

the safety of the vehicle and passengers is assured even if the system can no longer continue to operate. Although there are regulations which ensure that the safety decisions must be taken only by the driver (Hoppe et al., 2009), the situation has to change to reflect the rise of the autonomous and self-driving technology. In the meantime, a semi-automated response which requires partial interaction with the driver could be adopted.

Dynamic: The IRS may need to react dynamically. The system must react to the same attack in a different way based on many factors, such as the contextual information during the different operational modes of the vehicle, the attack severity, the targeted subsystem and other systems dependent on it. However, expressing all these different specifications using static security rules or decision table is not a trivial task. Therefore, the use of a dynamic security policy to express these parameters could be a preferable solution (Debar et al., 2007b; Hamad et al., 2017).

Deactiveable: A response to a malicious attempt, like any other security countermeasure, comes with a price. The IRS introduces extra overhead on the system's performance, based on the type and duration of the response. To minimise this overhead, the IRS should be able to deactivate the response mechanism when specific, predefined conditions within the security policy are met (Kanoun et al., 2013).

Distributed: Each ECU should have its own IRS. When a component is potentially under attack, the corresponding IRS detects the malicious attempt and decides locally on how to deal with the attack. Although this decision may respond to the attack against the specific component, it may not be optimal for the overall system. This is why a global response strategy may need to be implemented which is enforced by each individual component.

Efficient: Activating the response action should take place immediately after the attack is detected. In a different case, if a significant amount of time passes, the damage might already be done, negating the purpose of existence of the IRS. Therefore, the proposed IRS should evaluate the security policy as well as any other inputs in a very efficient way.

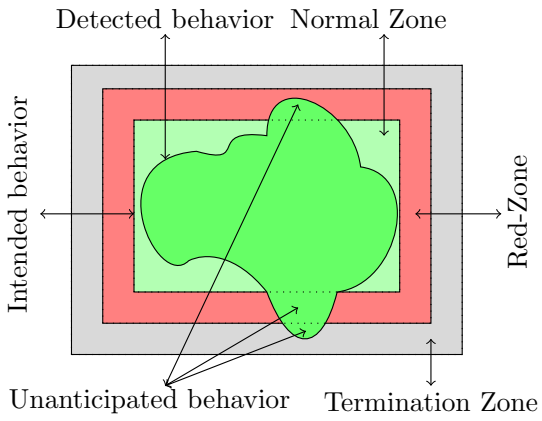

Figure 1: Red-Zone Principle

4 RED-ZONE FRAMEWORK

Improving the response strategies of any secure mechanism is dependent on early detection (i.e. early prediction) of a potential attack. Early prediction provides the system with a sufficient amount of time to initiate recovery actions to respond in an effective manner. Our strategy to solve this conundrum involves using the principle of Red-Zone which we introduced in (Prevelakis and Hamad, 2015).

4.1 Red-Zone Principle

Every task, in a given system, is usually designed to accomplish a certain job and behave in a well-defined way. The task is considered *normal* as long as it keeps complying with the predetermined execution profile. However, at run-time, a task may not exercise all the intended functionality and, in some circumstances, it may stray outside its planned operational profile and become *abnormal*, for many reasons, such as it is under attack, it includes a bug, or it behaves legally but the security policy is incomplete. The sudden exchange between the normal and abnormal state of the task may leave the system without sufficient time to recover which, in critical-safety systems, could lead to catastrophic consequences.

Therefore, instead of having only the two states for any task (i.e. normal and abnormal), we introduce a window of observation called the Red-Zone. The aim of this window is (a) to permit the task to overrun its designed operational profile until an ultimate limit, but in an observed mode and (b) to give the system a sufficient amount of time to initiate recovery actions while the task is in the Red-Zone and before it exceeds the ultimate limit, which then causes a security failure in the system. Figure 1 illustrates the possible

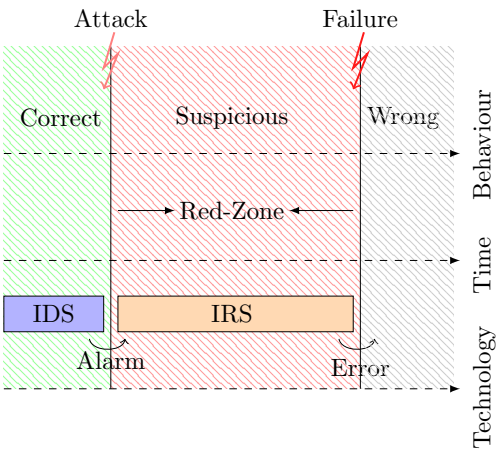

Figure 2: IRS based on Red-Zone Principle

behavior of a given task and how it could be placed within the different operational zones.

4.2 Red-Zone as an IRS

Figure 2 shows how we can use the Red-Zone principle as a base to develop an IRS and how it could be used to manage the interaction with the IDS. Based on this principle, the IDS is used to monitor and evaluate the behavior of the different vehicle software components within each ECU depending on predefined security policies. Each policy specifies the correct behavior of the given component. Any violation of this policy is used as a prediction of a potential security breach. At this point, the task enters the Red-Zone area and becomes suspicious. The activation of any response mechanism is triggered as soon as it receives an alarm about a detected potential attack from the IDS.

During the Red-Zone window, the IRS is responsible for activating multi-level responding strategies: (a) **Local** tactics to respond to the misbehaving component as well as other components on the same ECU, (b) **System-wide** responses which are implemented by an other IRS on a different dependent ECU within the vehicle. Such responses are activated as soon as a notification from the IRS where the suspicious component exists, is initiated. This alert includes all required information (such as the infected ECU, the attack details, and attack detection time) and transmitted using a predefined format similar to the format proposed in (Debar et al., 2007a). All these responses are defined within the security policy as we will see later. The local response of the malicious component can be:

- **Suspicious\Malicious Operational (S\MO)**: Although the task is considered malicious, the IRS leaves the task running until a certain limit. While it is executing, the component could be isolated and its communication with the other components on the same ECU or with other subsystems becomes very limited and traced. The logged information can be used later to design a proper mitigation mechanism. Moreover, the trust level of the component is degraded, so other components become more cautious about the information delivered from this component.
- **Suspicious\Malicious Silent (S\MS)**: The IRS kills the process whenever it exceeds the ultimate bound since the component will suffer from a security failure at this point (Avizienis et al., 2004). Although the IRS may stop the component at the end, which could be the goal of the attacker by causing a denial-of-service, the IRS has plenty of time to activate other response strategies ahead of this point.

Other responses could be also adopted by the IRS which will affect the whole system's configuration. Here we summarize some of these possible responses:

- **Reallocation**: Ensuring the availability of the different components is a high priority. The IRS can implement a response which ensures that a new healthy instance of the malicious component is initiated. The new instance is mapped to another ECU, thus other components that depend on the misbehaving one need to communicate with the newly initiated component instead. This requires the use of a new communication configuration to adjust the reallocation.
- **Degradation and Propagation**: Some less critical services which are mapped to the same ECU where the malicious component is running on, will have to become silent\inactive to save resources for the system to audit and to recover. Consequently, the same strategy will have to be applied to other interrelated ECUs which include components dependent on the silent ones.
- **Preparedness**: The IRS applies very strict restrictions on the ECUs which host the malicious component. This could include performing more security auditing operations on the communications of this ECU to prevent any stepping stone attacks. In case the attack was propagated across the whole ECU, total isolation of the ECU could be adopted.
- **Combination**: The IRS could adopt a combination of the aforementioned tactics to achieve the best possible protection before delivering the control to the human actor.

5 IRS DEVELOPMENT

Developing the response strategies to cover the large number of distributed components is a difficult task, requiring detailed knowledge of possible interaction paths and the dependencies among all those components, as well as the security threats that could target them along with the possible attack scenarios. When considering an evolving system, which shall be updatable (whereby component interactions may change), this task becomes even more difficult and requires a framework which offers IRS support under concurrent change. Therefore, the planning for the response strategies has to be integrated through the design and life-cycle of the system (i.e V-model).

In this section we show how to integrate the development of the IRS within the whole development process of the vehicle system. In addition, we discuss the various aspects of the system which affect the design and implementation of the IRS. Figure 3 shows the general scheme to develop the IRS. It depicts the interaction with the existing IDS system, as well as the role of the security policy which is used to specify the appropriate response. Here, we explain the different aspects which need to be considered during the development of the different responses.

5.1 Threat Model

Threat modeling is a systematic approach for describing and classifying the security threats which affect a system. Moreover, it provides significant information that would help to safeguard the target system and to develop effective response strategies against any attacks. During the design stage of the V-model, security requirements are defined to address all existing vulnerabilities identified by the threat model. At the same time, the response actions for violating these requirements are defined in an abstract view. At a later stage (i.e. implementation and integration) the response actions become more detailed to reflect the final system architecture and keep complying to the initial actions.

Generally, an attacker aims to violate one or more of the next requirements:

- Integrity: unauthorized change of the software component itself, the run-time environment, saved data, or exchanged messages among the different components.
- Confidentiality: unauthorized disclosure of the content of data (i.e. saved data or exchanged messages) by unauthorized actors.
- Availability: loss of the ability to reach a component or service as a result of malicious actions.

In addition, threat modeling provides us with a good comprehension about the prospective attackers, their capabilities and their objectives. This information is used by the attack tree to perform a risk assessment of each attack which affects the selection of the response strategy based on the attack severity.

5.2 Attack Tree

As we mentioned above, the main goal of the attackers is violating the security requirements via different tools and scenarios. An attack tree (Schneier, 1999) is used to investigate most of these scenarios in a tree structure as shown in Figure 3. The root of the tree represents the attacker's essential goal (e.g. Denial-of-service against one component), while the intermediate nodes of the tree (sub-goals) define different stages to achieve this goal. Each node in an attack tree could require achieving all of its sub-goals. In this case, the sub-goals are interconnected with an AND gate, while an OR gate is used when an upper node requires achieving at least one of its sub-goals. Leaf nodes represent atomic attacks. Attack scenarios are generated from the attack tree by traversing the tree in a depth-first method. Each attack scenario contains the minimum combination of leafs. The success of an attack depends on the subsequent success of interrelated attack steps.

The attack tree is created off-line for each system asset (e.g. hardware, software, data). Then, the defined attack trees are used to evaluate the security risks, to calculate the probability of a successful attack and to measure the defense (local response) cost (Edge et al., 2006). Calculating these metrics depends on different aspects (ISO/IEC 18045:2008, 2008), as well as the information provided by the threat model. The attack information, which is delivered by the IDS to the IRS after detecting the security violation, is used as an input for the IRS to choose the appropriate response.

5.3 Dependency Analysis

Another important aspect of implementing the response mechanism is to determine all dependency relations between other components and the malicious one. We call a task *dependent* on another task when it uses a service provided by that task. These relationships can be denoted in a dependency tree (Toth and Kruegel, 2002). Then the direct and indirect dependencies are specified. Moestl et al. (Moestl and Ernst, 2015) have proposed a cross-layer dependency analysis to detect dependencies between the different component across the different architectural model lay-

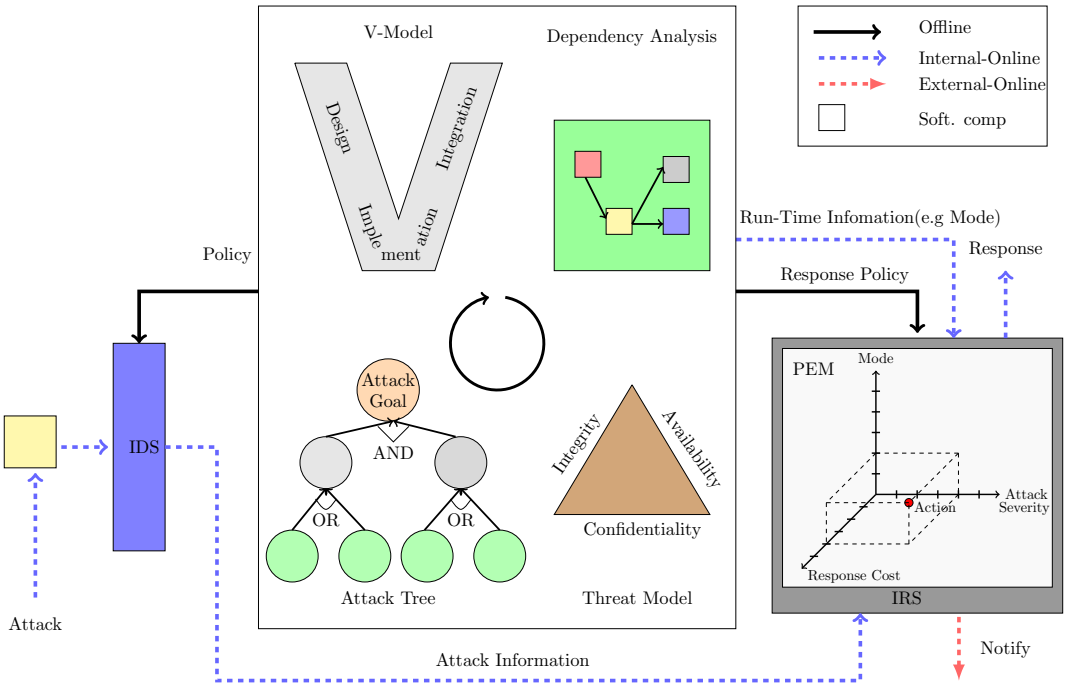

Figure 3: General Scheme of IRS Development

ers in the safety-critical systems such as a vehicle. By defining the dependencies, the IRS on the local platform can notify all other dependent sub-systems about the attack, thus response mechanisms on other platforms are also deployed. In addition, the system-based response cost can be evaluated by using the dependency analysis.

5.4 Security Policy

A security policy is used to define the response of the system when a task enters the Red-Zone, the conditions under which a Red-Zoned task may return to normal state and, finally, the response to a task breaching its ultimate limit. The main challenge in this context is to define a comprehensive security policy to manage the various responses in the intra-vehicle system after the final integration phase in a development process. This challenge originates from the high number of integrated ECUs in the vehicle, the complexity of the dependencies between the different components on each ECU and the various actors who participate the development process of these components. On the other hand, another challenge comes from the need to update the response strategies in a later stage to reflect the changes on the actual system architecture after the system update or upgrade. Therefore, the response policy is developed in an abstract way and is gradually detailed during the different development phases (design, implementation, integration) similar to (Hamad et al., 2017). The defined policy is separated from the service itself which facilitates the independent update of both.

Listing 1: Security Policy Structure

```
if (property == normal)
  do (allow)
else //Red-Zone
  if (property == malicious)
    if (attack_severity== high)
      if(mode == startup)
        do(response)
  else //fault
    do(response)
```

Listing 1 shows the structure of the security policy. Based on a certain property of a component, such as execution time, power consumption, system call distribution or bit-rate of message exchange, the boundary of the Red-Zone is determined in the policy and the response action within each zone is defined. Other properties such as the system operational modes (e.g. vehicle at startup mode) as well as the attack severity could affect choosing the response action. All this information is collected while the system is running and evaluated using the Policy Evaluation Module (PEM) to choose an action dynamically.

6 USE-CASE

Autonomous driving has seen huge progress and interest in recent years. Figure 4-(left) shows the typical autonomous vehicle system overview where many functionalities are collaborating and many sensors and actuators are involved. Figure 4-(right) shows the final integration of these functionalities on the actual ECUs. One important functionality within this architecture is the vehicle localization functionality which aims to give the vehicle the ability to determine its position with respect to the environment. The main information resource for this functionality is provided by the Global Position System (GPS). By looking at this functionality specifically, we want to show how we could develop response strategies when a security violation occurs against it and discuss all the aspects presented in Section 5.

In the first stage, threat modeling is performed and the security requirements for the localization functionality and other ones are defined: (a) availability: the vehicle needs to be always aware of its current position and (b) integrity: other system components which use the position information need to be sure that the data was received from an authenticated component and the position data has not been tampered with. Any violation of these requirements demands the adoption of the proper responses. Attack trees are developed to identify all the attack scenarios which could violate these two requirements. Attacks such as spoofing data attack, jamming attack, and many others are defined (Carroll, 2003; Nighswander et al., 2012). Based on the attacks details, detection mechanisms are defined and stated in the security policy to detect those attacks while the system is on-line.

The responses to these attacks also are expressed within the security policy. One response is receiving the service from anther source. Inertial Navigation System (INS) could be used as a temporary resource for providing localization information. Another response option is to reallocate the GPS module (in case of the Receiver Software Attack (Nighswander et al., 2012)) either within the same ECU or on another one (ECU_4 on our use-case). Both strategies require the communication reconfiguration for the direct dependent components. Dependencies analysis shows that *Local.* component, which is mapped on ECU_5, directly depends on the GPS module. Therefore, the security policy related to *Local.* component on ECU_5 should enable two communication paths with ECU_2 and ECU_4 whenever the *GPS Module* is behaving maliciously. It can also disable the link with ECU_1 to prevent the attack propagation, while *HMI* component, which is mapped on ECU_3 and indirectly dependent on the *GPS module*, can keep displaying the given data on the map, but with an alarm sign activated to express the low level of GPS data trust as shown in Figure 4-(right).

Implementation: We implemented a prototype of our IRS on a single platform. We used a Raspberry Pi 3 (RPI) running mickrokernel operating system to give us more capabilities to control the communication among the different components on the same platform as well as the isolation mechanism for the tasks. We used KeyNote policy definition language (Blaze et al., 1999) to express the security policy and Keynote Trust management to implement the PEM which is used to evaluate these security policies at run-time.

We measured the required time for choosing the proper response when one component becomes malicious. This time includes the evaluation time of the security policies of all dependent components in the same platform. Figure 5, shows the measured time when 1 to 5 components are involved.

7 DISCUSSION AND CONCLUSION

We propose to use the Red-Zone principle as a base to develop an IRS framework. Whenever a component is under a cyber-attack, the IDS notifies the IRS to be activated. The IRS uses the Red-Zone time period to implement response strategies. Based on this design, our proposed IRS meets the **proactive** requirement. There is a period before the task reaches its ultimate bound and causes any harm. How long this period is, is based on the selection of the boundary of the nominal behavior as well as the monitored property. In our previous work (Hamad et al., 2018), we have shown how the temporal property can be used as an indication to predict an attack at an early stage, before violating the ultimate temporal boundary (i.e. deadline). However, false positives can be a side effect when increasing the Red-Zone time window.

During this Red-Zone period, the IRS is able to activate a first level of response. We have already implemented many response strategies (Hamad et al., 2018) e.g. system notification. The IRS can, subsequently, apply a second set of actions such as placing the task under supervision. Another level of response can be applied when the task enters the termination zone (i.e. system failure). All these different response venues attest to how our system can be **het-**

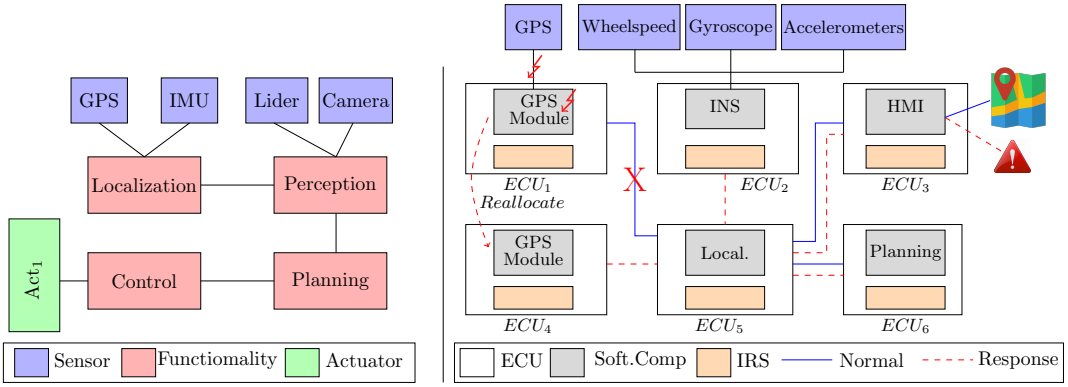

Figure 4: Response strategies in the presence of security attack on localization functionality of autonomous vehicle use-case.

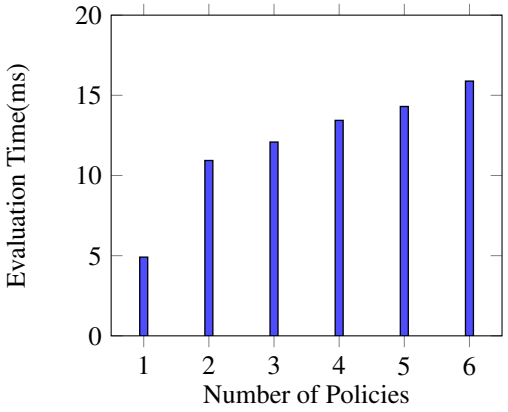

Figure 5: Required evaluation time for dependent components on the same RPI

erogeneous and **active** complying with the respective requirements.

The proposed framework is integrated within each ECU in the system (i.e. **distributed**). Our preliminary measurement results have shown how even resource-limited platforms are (**efficient**) to implement such a framework. IRSs are designed to use a predefined security policy in different strategies. We have extended the security policy, which was used in our previous work (Hamad et al., 2017) to include these strategies. We use the policy evaluation framework to evaluate the policies and to enforce the chosen response.

The role of the security policy is to define the proper response (**correct**) of the system when a task enters the Red-Zone, based on (*i*) the system states as well as the attack severity which could be calculated from the attack metrics in conjunction with a reference to the attack (**dynamic**), (*ii*) the conditions under which a Red-Zoned task may return to the normal state (**deactivatable**) and, finally, (*iii*) the response to a task breaching its ultimate limit.

As future work, we plan to concentrate on implementing system-wide IRS on multiple RPIs and experiment further in order to get more complete and comprehensive measurements to check the required time for the whole system to recover. The communication protocol between the different IRSs is another open topic for further research.

ACKNOWLEDGEMENTS

This work is supported by the DFG Research Unit Controlling Concurrent Change (CCC) project, funding number FOR 1800. Additionally, it is supported by the European Commission through the following projects: project H2020 THREAT-ARREST under Grant Agreement No. 786890, project H2020 CIPSEC under Grant Agreement No. 700378 and project H2020 CONCORDIA under Grant Agreement No. 830927.

REFERENCES

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33.

Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A. (1999). The keynote trust-management system version 2. Technical report.

Broy, M. (2006). Challenges in automotive software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 33–42. ACM.

Carroll, J. V. (2003). Vulnerability assessment of the us transportation infrastructure that relies on the global positioning system. *The Journal of Navigation*, 56(2):185–193.

Charette, R. (2009). This car runs on code. http://www.spectrum.ieee.org/feb09/7649.

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al. (2011). Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*.

Debar, H., Curry, D., and Feinstein, B. (2007a). The intrusion detection message exchange format (idmef). Technical report.

Debar, H., Thomas, Y., Cuppens, F., and Cuppens-Boulahia, N. (2007b). Enabling automated threat response through the use of a dynamic security policy. *Journal in Computer Virology*, 3(3):195–210.

Edge, K. S., Dalton, G. C., Raines, R. A., and Mills, R. F. (2006). Using attack and protection trees to analyze threats and defenses to homeland security. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–7. IEEE.

Hamad, M., Hammadeh, Z. A., Saidi, S., Prevelakis, V., and Ernst, R. (2018). Prediction of abnormal temporal behavior in real-time systems. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 359–367. ACM.

Hamad, M., Nolte, M., and Prevelakis, V. (2016a). Towards comprehensive threat modeling for vehicles. In *1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems (CERTS)*.

Hamad, M., Nolte, M., and Prevelakis, V. (2017). A framework for policy based secure intra vehicle communication. In *Vehicular Networking Conference (VNC), 2017 IEEE*. IEEE.

Hamad, M., Schlatow, J., Prevelakis, V., and Ernst, R. (2016b). A communication framework for distributed access control in microkernel-based systems. In *12th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT16)*, pages 11–16.

Herold, N. (2017). *Incident Handling Systems with Automated Intrusion Response*. Dissertation, Technische Universität München, München.

Hoppe, T., Kiltz, S., and Dittmann, J. (2008). Adaptive dynamic reaction to automotive it security incidents using multimedia car environment. In *Information Assurance and Security, 2008. ISIAS'08. Fourth International Conference on*, pages 295–298. IEEE.

Hoppe, T., Kiltz, S., and Dittmann, J. (2009). Applying intrusion detection to automotive it-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4(6):226–235.

Inayat, Z., Gani, A., Anuar, N. B., Khan, M. K., and Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*, 62:53–74.

Ishtiaq Roufa, R. M., Mustafaa, H., Travis Taylora, S. O., Xua, W., Gruteserb, M., Trappeb, W., and Seskarb, I. (2011). Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In *19th USENIX Security Symposium*.

ISO/IEC 18045:2008 (2008). nformation technology – Security techniques – Methodology for IT security evaluation. Standard, International Organization for Standardization.

Kanoun, W., Samarji, L., Cuppens-Boulahia, N., Dubus, S., and Cuppens, F. (2013). Towards a temporal response taxonomy. In Di Pietro, R., Herranz, J., Damiani, E., and State, R., editors, *Data Privacy Management and Autonomous Spontaneous Security*, pages 318–331, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kemmerer, R. A. and Vigna, G. (2002). Intrusion detection: a brief history and overview. *Computer*, 35(4):supl27–supl30.

Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al. (2010). Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE.

Larson, U. E. and Nilsson, D. K. (2008). Securing vehicles against cyber attacks. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, page 30. ACM.

Larson, U. E., Nilsson, D. K., and Jonsson, E. (2008). An approach to specification-based attack detection for in-vehicle networks. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 220–225. IEEE.

Le Lann, G. (1996). *The ariane 5 flight 501 failure-a case study in system engineering for computing systems*. PhD thesis, INRIA.

Locasto, M. E., Wang, K., Keromytis, A. D., and Stolfo, S. J. (2005). Flips: Hybrid adaptive intrusion prevention. In *International Workshop on Recent Advances in Intrusion Detection*, pages 82–101. Springer.

Markit, I. (2016). Vehicles getting older: Average age of light cars and trucks in u.s. rises again in 2016 to 11.6 years, ihs markit says. https://news.ihsmarkit.com/press-release/automotive/vehicles-getting-older-average-age-light-cars-and-trucks-us-rises-again-201.

Moestl, M. and Ernst, R. (2015). Cross-layer dependency analysis for safety-critical systems design. In *ARCS 2015-The 28th International Conference on Architecture of Computing Systems. Proceedings*, pages 1–7. VDE.

Nadeem, A. and Howarth, M. P. (2014). An intrusion detection & adaptive response mechanism for manets. *Ad Hoc Networks*, 13:368–380.

Nighswander, T., Ledvina, B., Diamond, J., Brumley, R., and Brumley, D. (2012). Gps software attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 450–461. ACM.

Prevelakis, V. and Hamad, M. (2015). Extending the operational envelope of applications. In *8th International Conference on Trust & Trustworthy Computing (TRUST 2015)*.

Schneier, B. (1999). Attack Trees - Modeling security threats. *Dr. Dobb's Journal*.

Shameli-Sendi, A., Ezzati-Jivan, N., Jabbarifar, M., and Dagenais, M. (2012). Intrusion response systems: sur-

vey and taxonomy. *Int. J. Comput. Sci. Netw. Secur*, 12(1):1–14.

Stakhanova, N., Basu, S., and Wong, J. (2007). A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1-2):169–184.

Sterne, D., Djahandari, K., Wilson, B., Babson, B., Schnackenberg, D., Holliday, H., and Reid, T. (2001). Autonomic response to distributed denial of service attacks. In *International Workshop on Recent Advances in Intrusion Detection*, pages 134–149. Springer.

Strasburg, C., Stakhanova, N., Basu, S., and Wong, J. S. (2009). A framework for cost sensitive assessment of intrusion response selection. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, volume 1, pages 355–360. IEEE.

Toth, T. and Kruegel, C. (2002). *Evaluating the impact of automated intrusion response mechanisms*. IEEE.

Tryfonas, T., Gritzalis, D., and Kokolakis, S. (2000). A qualitative approach to information availability. In *IFIP International Information Security Conference*, pages 37–47. Springer.

Vöelp, M. and Esteves-Verissimo, P. (2018). Intrusion-tolerant autonomous driving. In *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, pages 130–133.