Practical Information-Flow Aware Middleware for In-Car Communication

Alexandre Bouard
BMW Forschung und Technik Gmbh
Munich, Germany
alexandre.bouard@bmw.de

Benjamin Weyl
BMW Forschung und Technik Gmbh
Munich, Germany
benjamin.weyl@bmw.de

Claudia Eckert
Technische Universität München
Garching, Germany
claudia.eckert@in.tum.de

ABSTRACT

Today's vehicles are increasingly connected to Internet, devices and integrate more and more electronic components. More than just ensuring their passengers' safety, which remains nevertheless one of their main objectives, cars have to deal with private information and encounters the same security issues as traditional computers. Until recently, automotive technologies allowed very little space for security, but the transition towards full Ethernet-based on-board network will change this situation. In this paper, we present solutions for decentralized information flow control in order to enhance the security and privacy level of the car data management. We describe the implementation of these mechanisms in an automotive middleware and propose its evaluation.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Car, Security & Privacy, Decentralized Information Flow Control, Middleware, Automotive Applications

1. INTRODUCTION

During the last couple of decades, cars have evolved from a simple mean of transportation to a complex distributed electronic system, simultaneously managing infotainment and safety functionalities. Today, while still serving their primary purpose, cars provide new mobility services taking advantages of powerful embedded hardware platforms and efficient interconnections with the external environment (e.g., Internet, smartphones, other cars). As a consequence, vehicles will have to process more and more data from different sources, belonging to different protagonists and eventually, like our smartphones, they will soon allow the user to install third-party applications (TPAs) [1].

CyCAR'13, November 4, 2013, Berlin, Germany.

http://dx.doi.org/10.1145/2517968.2517969.

In the last few years, information about several weaknesses related to automotive security and the feasibility to easily exploit them [2] circulated and have been discussed by the research community. However, current technologies' limitations and requirements for low latency and robustness left little room for security. Part of the solutions seems to lie in the use of Ethernet and the Internet Protocol (IP) as standard communication protocol for the on-board network [3]. While providing a bigger bandwidth and strong security protocols for encryption and authentication, it does not completely address every information security issue. Increasing complexity of the car software and the integration of untrusted components like consumer electronics (CE) devices or TPAs could result in data leakage and even more exploitable interfaces.

In order to improve security and privacy on the on-board network, we base our approach on a Decentralized Information Flow Control (DIFC) model [4]. This approach allows control of the pieces of information which can be exchanged between two entities and to monitor how they are spread in the system. The goal is to have the car acting as a data safe and regulating every internal and external data exchange according to specific security policies. A trusted computing base including a part of the software, the middleware, is in charge of enforcing these rules and allows a homogeneous and scalable way to manage the authorization process.

The main contributions presented in this paper are:

- An *authorization model* enforcing DIFC for on-board communications and allowing a secure integration of CE devices and TPAs.
- A *framework implementation* integrating the automotive IP-based communication middleware *Etch* [5] and isolation techniques for the TPAs.

The rest of the paper proceeds as follows. After having given a brief overview about the on-board automotive architecture and related work in Section 2, Section 3 introduces the model used to apply DIFC in an automotive environment and its enforcement. Finally Section 4 presents our implementation and Section 5 provides its evaluation.

2. BACKGROUND AND RELATED WORK

This section provides background information on future automotive systems and related work about security and privacy. A threat model and some relevant scenarios are described as well.

2.1 Future Automotive architecture

The automotive on-board network interlinks up to 80 Electronic Control Units (ECUs) thanks to different communication buses and is organized in domain-specific sub-networks (e.g., for power train or infotainment). On-board applications include several elementary function blocks distributed over separated ECUs exchanging broadcasted signal-based messages. Due to on-board plaintext communication and a lack of input validation in the ECUs, cars have been shown weak against common security exploits, e.g., packet injection, buffer overflow [2].

The advantages of Ethernet/IP for vehicle on-board network are twofold. Firstly, a bigger bandwidth will allow to internally exchange larger objects (e.g., environment models) between ECUs and to comply with the future application requirements for driver assistance and infotainment [6]. Secondly, mature and secure protocols from the Internet world will be immediately usable. Besides, the development of engineering-driven middleware will greatly simplify the communication management. It will abstract and automate the network addressing and security enforcement [7]. Then the centralization of most of the external wireless interfaces (e.g., LTE, Wi-Fi) in a multi-platform antenna-ECU (called proxy here) will enable car makers to design a single security gateway for all Car-to-X (C2X) communications [8].

2.2 Threat Model and Scenario

Today's cars are facing several challenges: their functional behavior relies on complex software processing a considerable amount of sensitive data and soon integrating TPAs. Defects in the application logic or weak security mechanisms [2] could result in leaks of private information or industrial secret and in the worst case endanger the car safety mechanisms and potentially the life of the car passengers. The car IT system should enforce security policies and consistent mechanisms assuring car-wide confidentiality and integrity guarantees.

Our scenario, depicted in Figure 1, features a TPA, running on the Head Unit (HU) and a CE device communicating with several internal components of the car. This work aims at improving information security in cars and at addressing the threats related to unauthorized entities wanting to access sensitive automotive data. Our threat model includes unintentional programming bugs at the application level causing information leakages and unfair authorized parties, internal ones like TPAs or external ones like CE devices, trying to elude security policies, i.e., by trying to access or leak data they have no authorization for.

Assumptions: Future ECUs will make use of a security middleware and will communicate over strong security protocols like IPsec [7]. Additionally they will soon integrate a hardware secure extension providing secure key storage and secure boot [9]. Besides the development of an automotive middleware layer according to secure coding practices will also limit the risk of attacks involving overwriting of stack pointers, e.g., buffer overflow. Consequently we assume that the middleware and the hardware platform cannot be compromised. We trust the ECUs to establish secure communications channels with each other and to enforce the expected security mechanisms. Denial-of-Service (DoS) attacks are partly considered at the end of this work.

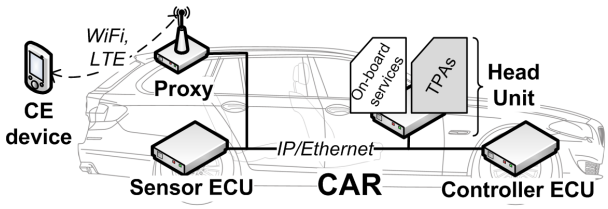

Figure 1: Automotive scenario. Solid right-angle lines represent the wired on-board network. The dashed arrows represent external wireless communications. We define as service, a group of on-board applications sharing a same middleware. (TPA: Third-Party Application)

2.3 Related Work

Information Flow control (IFC) is a type of mandatory access control. Principals (e.g., a person) having access to an object (e.g., a document) and the aforementioned object are given security labels designating a clearance level (e.g., secret, top-secret). A partial order, based on the labels, defines whether the access is granted [10]. DIFC [4] extends the IFC model. It allows an object (e.g., an application) to divide its rights by creating new labels and to control with more flexibility the label management and enforcement. The model has been adapted at the granularity of a process. During runtime processes are separated between trusted and untrusted. Label-based rules are enforced locally by the operating system (OS) [11, 12] but labels can be exchanged and enforced between hosts over the network as well [13]. While clearly enhancing the information security, these approaches are too fine-grained and suffer from a too significant performance impact, we therefore chose a lighter approach and enforce DIFC only on communication between on-board services. Enforcing IFC thanks to label exchanges is not new. The DEFCON customized Java virtual machine can enforce isolation and label management when all the applications are present on a single host [14]. For Pedigree, a central server and customized network switches can distribute and enforce IFC policies on communications between several hosts in a corporate network [15]. In contrast, the software components of a car are distributed over several hardware platforms equipped with different OSs. In order to reduce the risk of errors, latency and maintenance complexity, the IFC enforcement cannot rely on any central entity for the authorization management and cannot be enforced by the hardware or its OS.

3. DIFC MODEL FOR AUTOMOTIVE ON-BOARD NETWORK

Automotive DIFC is about monitoring the in-car propagation of data of interest. But instead of monitoring every process, we focus on network exchanges between on-board services and external devices. Services regroup several on-board applications on top of a same middleware layer The applications of a service share the same security concerns for integrity (e.g., because they trigger the same safety mechanisms) and confidentiality (e.g., because they share data with the same sensitivity). For this reason, a label is assigned to every service/device perpetrating a message exchange. A trusted part of the ECU software, the middle-

ware, which is independent from the labeled resources and whose integrity can be checked, is in charge of monitoring and labeling network exchanges. The rest of the section describes our label-based security model, adapted from [13, 14].

3.1 Security Labels

One security label is assigned to each principal. Comparing labels allows to constrain the information flow and therefore to protect the information integrity and confidentiality, for example by isolating potentially corrupted data from critical applications or preventing unauthorized disclosure of private information.

Labels comprise two components: the first characterizing the principal's secrecy S, the second its integrity I. S and I are two sets of tags. A tag represents the concern of an individual about the secrecy/privacy (in S) and integrity (in I) of some data. Tags are unique values in the system, implemented as bit-strings, we refer to them with symbolic names, like x_s. The subscripts s and i of x_s and x_i designate, respectively a secrecy- and an integrity-tag. The x specifies the principal, whose security concerns it characterizes, i.e., the service x or the CE device of the driver x. Secrecy tags are "sticky": once added to a piece of information, it cannot flow to a principal lacking the exact same tag. On the other hand integrity tags are fragile: a piece of information loses it as soon as the principal processing it is differently tagged.

The labels form a lattice enforcing a form of mandatory access control. Information labeled with the secrecy tags of S_A can flow to a principal labeled with S_B if and only if the tags of S_A are included in S_B. Inversely, information labeled with the integrity tags of I_A can flow to a principal labeled with I_B if and only if the tags of I_A contain the ones of I_B. We define the partial order "$\prec$" (pronounced "can flow to") for two labels $L_A = (S_A, I_A)$ and $L_B = (S_B, I_B)$ as:

$$L_A \prec L_B \text{ iff } S_A \subseteq S_B \text{ and } I_A \supseteq I_B$$

Because the services are distributed over different ECUs and do not know each other's labels, we label the exchanged messages. When A sends a message M to B with L_A, L_M, L_B their respective labels, we enforce the property $L_A \prec L_M \prec L_B$. Constraining the message label allows the message to be disclosed by A (A can label M with L_A such that $L_A \prec L_M$) and then accepted by B (B checks the condition $L_M \prec L_B$, i.e., $L_A \prec L_B$). In our scenario, data stored on the HU should be tagged with different values reflecting the different drivers' secrecy, so that only an appropriate TPAs or CE devices can receive a message containing a particular driver's data.

3.2 Tag Ownership

If information could only follow the partial order "$\prec$", the labeled messages would only be transmitted to principals classified at a greater or equal level of secrecy and most data would never be able to leave the car. DIFC decentralizes the management of exceptions: each service S may be assigned a set of tags O, allowing it to omit from the label restrictions included in O. We say that S owns the tag of O. Obviously no service should own all the tags of the system; a service should own only the necessary tags in order to remain functional.

We write "$\prec_O$" (pronounced "can flow to, given O") the new partial order taking into account the ownership to the tag of O. Practically, a tag t included in O confers the possibility for a service S to omit the restriction imposed by t. For data flowing from A to B, except for the tags included in O, the label L_A contains all the integrity tags of L_B and L_B contains all the secrecy tags of L_A. We now define "$\prec_O$" for two labels $L_A = (S_A, I_A)$ and $L_B = (S_B, I_B)$ and the ownership O as:

$$L_A \prec_O L_B \text{ iff } S_A - O \subseteq S_B - O \text{ and } I_A - O \supseteq I_B - O$$

Like earlier, A with the ownership O_A can send a message M and B with the ownership O_B can receive it if and only if $L_A \prec_{O_A} L_M \prec_{O_B} L_B$. An untrustworthy TPA will not be given any ownership and therefore will not be able to modify its own label in order leak the driver's data or access other driver's data. On the contrary the proxy will be given the ownership of the driver's tags in order to be able to send his data to his CE device. The proxy provides a high security level and is trusted to use its ownership in a secure manner.

During runtime, a principal may decide to create and own a new tag, which reflects a new security concern. At its discretion, it can decide to grant the ownership to another principal. Thus the proxy can create for every new user new tags for secrecy and integrity and grant their ownership to the HU, in order to label and protect the users' data the HU may store.

Dynamic Label Assignment (DLA): A DLA is an explicit request from a service to another in order to add a tag to the latter's label. A newly installed TPAs starts with no tag ownership and an empty label, preventing it from receiving and contacting most on-board functions. In order to send to it some driver's d private data, the HU, which owns d_s can force the TPA to include them in its label. The tags cannot be taken out and constrain the TPA to only send messages to principals including d_s in their label.

3.3 DIFC Framework Architecture

In order to maintain performance and to limit the risk of error, we only monitor on-board information flows between services. Applications in a same service share the same security concerns and therefore can be labeled together. Services are isolated from each other in their own address space or physically separated, i.e., on different ECUs. We chose to enforce DIFC at the middleware level, the software layer common to every service, easily auditable and in charge of the network communications. Our architecture is depicted in Figure 2. Applications in different services interact through their middleware, which provides the functional logic for communication and protocol implementation (Secure Channel Manager). The middleware header of every message is extended with a field containing the message label. The middleware labeler makes sure that the partial order $\prec_O$ between service and message labels is enforced for both incoming and outgoing traffic. Applications are DIFC-unaware and do not take part of the label management. The remainder of this section provides more specifications about the automotive label assignment, policies and management.

Label assignment: Each service x is labeled with its own integrity and secrecy tags (x_i, x_s) characterizing its own security concern. The assignment of additional label tags or tag ownership is defined by the car manufacturer at design time, depends on the implemented use cases and remain static along the car life. During runtime the proxy is the only one able to dynamically create new tags related to a

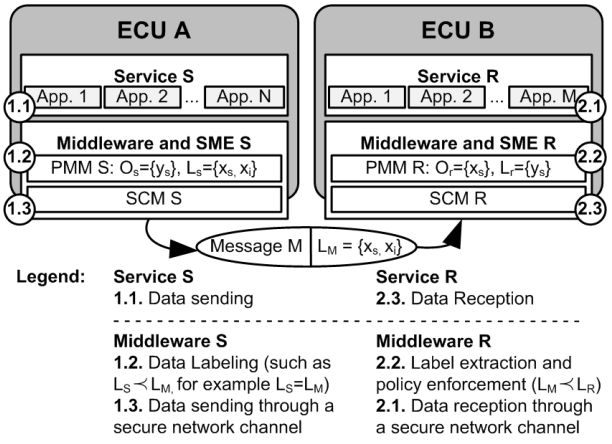

Figure 2: Overview of the framework architecture. A labeled message M is exchanged between 2 applications (App.) of S and R services through a secure channel. L and O designate the service label and ownership. x_s, y_s and x_i are secrecy/integrity tags.

new user profile and to grant them to relevant ECUs, like the HU. We do not consider the addition of new services and therefore of new service tags during runtime. But obviously, new service tags would force the car maker to update the middleware label and ownership of every middleware communicating with the new service.

Security Policies: We identify two types of policies. The first ones are static and enforced in the Secure Channel Manager for establishing communications channels between ECUs, e.g., in the database listing every IPsec Security Association. The second ones, enforced in the labeler, specify which operations on labels are authorized, and specify the enforcement of $\prec_O$ for incoming and outgoing message labeling. These rules are defined by security experts during the design phase and remain static along the runtime.

Label Management In order to control on-board information flows, on-board services exchange labeled messages and store labeled data. However, in case of complex data fusion involving different labels, we chose to not concatenate the labels. Instead we define simple prioritization rules, which allow the middleware to restrain the size of the message labels, i.e., one tag for the secrecy and one tag for the integrity. For example, the secrecy tag of a sensitive service has priority over a user tag, i.e., the data, which are sensitive for the car manufacturer, should stay in the car. The tag of user has priority on a non-sensitive service tag. This limitation of the label complexity allows to keep the DIFC enforcement efficient. The list of sensitive services is embedded in the middleware and statically setup by the car manufacturer.

4. IMPLEMENTATION

This section describes the DIFC extensions of the middleware *Etch* and its associated security communication proxy.

Middleware: We made use of the middleware *Etch*, an open-source software project under the Apache 2.0 license. *Etch* offers a modular and extensible architecture providing an efficient serialization and is a serious candidate for automotive purposes [17]. We chose its C-binding and extended the middleware header with two fields of 15 bytes for the integrity and confidentiality labels. Label serialization/extraction and enforcement is performed in the software logic of the middleware. The tags of the service label and ownership are stored in 2 distinct tables. An implementation making use of hash table may provide better performance for a big number of tags. The application part can consult the labels, but not modify them. A future version of *Etch* will allow to directly specify labels and ownerships of the service through a suitable interface description language (IDL) and will provide automatic insertion of the code snippet enforcing the label policies in the middleware code.

We developed an *Etch* proxy in C, similar to the one developed for [8]. The proxy provides two secure communication interfaces: external over SSL and internal over IPsec. Internal and external communication partners communicate over a mirror-service, making the communication decoupling totally transparent. The proxy is application-unaware. Either it extracts a label from the payload of an outbound message and enforces the required policy, or it adds a label to the inbound message. User tags are based on the identity provided by the client certificate of the external SSL connection.

TPA environment: Regarding the isolation of a TPA, we make use of the XEN® hypervisor 4.2, set up on the HU. We run the "trusted" HU middleware and applications, developed by the car manufacturer, in the most privileged domain, called Dom0. The untrusted TPAs run in unprivileged cells, called DomU. Communications between Dom0 and the DomUs occur over a virtualized bridge. XEN isolates and prevents the DomUs from disturbing Dom0 or from getting a direct access to the on-board network otherwise. The untrusted applications run on top of a label-unaware ETCH middleware. A dedicated part of the HU middleware running in Dom0 forwards messages toward and from the TPAs and enforces the DIFC policies for every message entering or exiting an untrusted cell.

Testing environment: For the implementation and the experiments described in Section 5.2 we used three computers interlinked with Gigabit Ethernet and running standard 32-bit Fedora Linux on an Intel Atom N270 (1,6 GHz) with 1GB RAM. The DomU runs a OS Debian 6.0 with 256MB of allocated RAM. While being more resourceful than many embedded platforms of the cars, they provide performances similar to a HU [18]. Besides we did not extensively modify the *Etch* middleware mechanisms, which provide suitable performances when tested on a microcontroller [17]. Therefore we believe that the addition of this simple DIFC access control layer should not significantly impact the system. Though this has to be verified for a more rigorous validation.

5. EVALUATION

In order to evaluate our system, we discuss the security of our concepts and quantify the performance overhead of our implementation.

5.1 Security Evaluation

In order to illustrate how our DIFC framework helps to build a secure on-board system, we focus on the scenario that reflects the threat model presented in Section 2.2. Figure 3 shows the overall scenario architecture and label distribution. A CE device connects to the proxy and can then contact on-board service functions and a TPA through the HU. The TPAs can get access to the driver's data stored

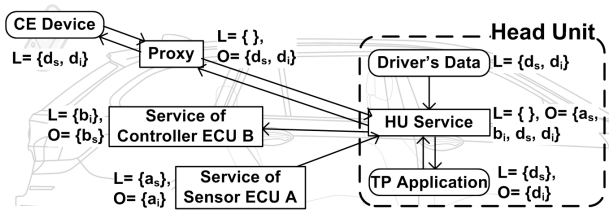

Figure 3: Automotive scenario. Rectangles represent services running on an independent middleware. Round boxes represent DIFC unaware applications, devices and files. Solid arrows represent middleware-based communications.

on the HU or data from a ECU A and potentially trigger mechanisms on the ECU B.

Discussion: Our architecture does not propose any hierarchy of privileges, on contrary all services are mutually distrustful. As a consequence the effects of a successful attack or bug in an application are limited to the labels their middleware is associated with.

First the CE device gets authenticated by the proxy, which binds the device to the driver's labels d_s and d_i. The proxy then explicitly grants the HU with these labels, so that the HU can label the driver's data and label the TPA as acting for the driver (DLA with d_s and d_i). CE device and TPA are untrustworthy components, we do not rely on them to handle any label, instead the proxy and a dedicated part of the HU service are enforcing the DIFC rules for them.

The TPA is minimally trusted and is confined to the tags d_s and d_i. The secrecy tag d_s constrains the TPA to read sensitive data, which only belongs to the driver. The ownership of the integrity tag d_i allows the TPA to write on the driver's data. The presence of d_i in its label would force the TPA to receive d_i labeled information and prevent it from accessing other nonsensitive information like configuration files. In our scenario, a malicious TPA is limited to send messages only to the driver's device and can only modify his data. A label only including d_s, without d_i or ownership of d_i, would limit its data access to "read-only". Because a CE device is bound to one identity and communicates with the TPA, we limit the label of the TPA to tags of only one person. In a same manner, the CE device is restricted to the tags of its owner. It can only contact services with no integrity tags in their label. Such services should be carefully designed and follow secure coding practices both at the middleware and application levels. Besides, the CE device can only receive driver-labeled or unlabeled, i.e., nonsensitive, information, which limit the risk of information leakage. The proxy can generate new tags for the devices it communicates with, owns them and is empty labeled, in order to always be able to communicate with any new device or Internet service.

Labels can also constrain information flows between services. The HU service can receive messages from ECU A only if it the HU has the tag a_s in its label or ownership. Therefore, even when forwarded by a multicast address by mistake, messages from A containing sensitive information will never been processed and sent out by the proxy or by another service without the tag a_s. On the other hand the HU owns the tags a_s and d_s, the HU can therefore make certain messages from A available to the TPA. An ECU B with b_i in its label, will only receive data (e.g., call to trigger a mechanism) from ECUs with b_i in their label or ownership.

No middleware is fully trusted and benefits from all tag ownership. Labels, ownership and DLA allow to express the security concerns of components presenting different levels of security. Each on-board service can specify its own security requirements and trust the remote middleware layer to enforce them.

System Limitations: Our framework relies on the integrity of the OS and middleware. Despite all the effort of car manufacturers to audit and test their software, vulnerabilities may allow an attacker to take control of the labeling process. The secure boot mechanisms, mentioned in Section 2.2, won't cope with runtime attack. Intrusion detection systems (IDS) may be one solution to detect and inform the driver about an ongoing attack. On the network, an IDS can monitor the traffic and ensure that the right services exchange the right message labels at a reasonable rate, i.e. not flooding the network. Within an ECU, an IDS can perform scans and recognition of instruction patterns. IDS solutions cause usually significant performance degradation and should be only used, if necessary, for non time-critical applications and subnets. A less costly solution, to limit the impact of a successful attack, is to limit the services' size, so that they only handle a little number of tags. Every message can be extended with an unforgeable token specifying which tags are included in the sender's label and ownership. The token has to be signed by a trusted entity and easily verifiable by the receiver and can prevent a compromised service to use any kind of tag.

Regarding the TPAs, they run isolated in an unprivileged XEN cell, where the corruption of the OS and the exhaustion of the virtual machine (VM) resources can be easily detected. Besides, the HU service can also restrain the emission rate of the TPA, in order to limit its capacity to launch a flooding DoS attack. However, several users, i.e., a driver and some passengers, imply to have several running VMs and may require too many resources from the HU. Another solution, not investigated here, would be to run the TPAs in a binary instrumentation framework. Such framework allows to taint sensitive data from different users within the running application. It can detect attacks like buffer overflows and determine, whether an output from the TPA has to be considered as sensitive or dangerous.

5.2 Performance Evaluation

We measure the middleware throughput (in calls/sec) between a CE device and an on-board TPA in order to demonstrate the overhead of our DIFC framework. Benchmarks are run on three separated machines running our Etch services: CE device, proxy and HU. Our setup here is similar to the scenario presented in Section 5.1. The CE device sends a simple Etch message containing an integer to the TPA. Based on this integer, the TPA retrieves a series of integers from a database of the HU, computes an answer and sends it back. This message exchanges pass through the proxy and the HU, where DIFC rules are enforced. Our results, in Table 1, present the throughput performances of this scenario for various security levels. We first measured them without any security feature enabled as reference (1). We then performed the same tests when adding the isolation of the TPA in a XEN cell (2) and the communication encryption (3) (SSL on the link CE device–Proxy and IPsec for

Table 1: Middleware throughput performance of the scenario presented in Section 5.2 (Virt.: TPA is placed in a Virtualization Environment, Enc.: The communications are encrypted and DIFC: DIFC rules are enforced). Factor (i) presents the normalized performance with (1) as reference, while factor (ii) takes (3) as reference.

Enabled Security Feature	Null (1)	Virt. (2)	Virt./Enc. (3)	DIFC/Virt./Enc. (4)
Throughput (call/sec)	230	196	152	137
Factor (i)	1	0.85	0.66	0.60
Factor (ii)	-	-	1	0.90

the link Proxy–HU), in order to determine a lower bound overhead imposed by the security framework without DIFC. We finally repeated the measurements when enforcing DIFC rules on the proxy and HU (4) and evaluated their impact.

Discussion: Our results, in Table 1, show that the virtualization environment and the use of security protocols are responsible for the most significant part of the performance loss (∼34%). The XEN paravirtualization and the virtual network bridge between Dom0 and a DomU decreases the system performance by 15%. The payload encryption/ decryption and integrity checks, performed by SSL for external communications and IPsec for internal ones, cause an additional penalty of 22%. The enforcement of label-based rules for communications between Proxy and HU and between HU and TPAs represents only 10% of the overhead. The overhead imposed by the DIFC rules are relatively small in comparison to the other security features presented here. The use of the DIFC framework seems to be suitable for infotainment use cases involving a CE device and requiring moderate bandwidth (130 calls/sec with payloads containing 32 bits of information). In addition, our measurements for communications between 2 nodes (ECUs), with DIFC and IPsec, have shown to reach 1100 calls/sec for small payload (32 bits) and 9,8 Mbit/sec when using bigger ones (65 kbit). Such results demonstrate that despite the security enforcement, our middleware remains functional, flexible and can provide relatively large bandwidth and a high frequency of calls. But, as previously said, our evaluation is mostly focused on our middleware in a 3-node network for a specific scenario. Additional investigations in larger network producing more traffic are recommended for further validation with use cases demanding large bandwidths (e.g., real time video streaming) and high robustness (e.g., safety function).

6. CONCLUSION

In this paper, we presented a new automotive security framework leveraging information flow control techniques to enhance the security level of the on-board data management. While we have certainly shown that these concepts can be efficiently adapted and integrated into an automotive middleware for infotainment applications, further investigations, in direction of system latency and robustness, are required to use it for safety-critical applications. Although DIFC alone cannot mitigate the risk induced by TPAs, carefully chosen isolation and monitoring mechanisms, complying with the automotive requirements, need to be added. As following work, we are now focusing on investigating the feasibility of tracking information flow at the binary-level in very untrusted applications.

7. REFERENCES

[1] Lutz Z.: Renault debuts R-Link, engadget and Renault press release at LeWeb'11 (2011)

[2] Koscher, K. et al: Experimental Security Analysis of a Modern Automobile. In *Proc. of the 31st IEEE S&P*, pp. 447–462, IEEE (2010)

[3] Glass, M., Herrscher, D., Meier, H., Piastowski, M., Shoo, P.: SEIS - Security in Embedded IP-based Systems. In *ATZelektronik worldwide, 2010-01* (2010)

[4] Myers, A. C., Liskov, B.: Protecting Privacy Using the Decentralized Label Model. In *ACM Transactions on Software Engineering and Methodology*, vol. 9, pp. 410–442, ACM (2000)

[5] Etch home: `http://incubator.apache.org/etch/`

[6] Schönenberg, P.: Introduction of Ethernet. In *6th Vector Congress* (2012)

[7] Bouard, A., Glas, B., Jentzsch, A., Kiening, A., Kittel, T., Weyl, B.: Driving Automotive Middleware Towards a Secure IP-based Future. In *10th escar* (2012)

[8] Bouard, A., Schanda, J., Herrscher, D., Eckert, C.: Automotive Proxy-based Security Architecture for CE Device Integration. In *5th MobileWare 2012*, pp. 62–76, Springer (2012)

[9] Fujitsu Semiconductor Europe: Fujitsu Announces Powerful MCU with Secure Hardware Extension (SHE) for Automotive Instrument Clusters. In Fujitsu Press Release at www.fujitsu.com (2012)

[10] Department of Defense: Trusted Computer System Evaluation Criteria In *Orange Book* (1983)

[11] Efstathopoulos, P. et al: Labels and Event Processes in the Asbestos Operating System. In *Proc. of the 20th ACM SOSP*, pp. 17–30, ACM (2005)

[12] Zeldovich, N. et al: Making Information Flow Explicit in Histar. In *Proc. of the 7th USENIX OSDI*, pp. 19–19, USENIX Association (2006)

[13] Zeldovich, N., Boyd-Wickizer, S., Mazières, D.: Securing Distributed Systems with Information Flow Control. In *Proc. of the 5th USENIX NSDI*, pp. 293–308, USENIX Association (2008)

[14] Migliavacca, M., Papagiannis, I., Eyers, D. M., Shand, B., Bacon, J., Pietzuch, P.: Defcon: High-Performance Event Processing with Information Security. In *Proc. of the USENIX ATC'10*, pp. 1–1, USENIX (2010)

[15] Ramachandran, A., Mundada, Y., Tariq, M. B., Feamster, N.: Securing Enterprise Networks Using Traffic Tainting. In *Special Interest Group on Data Communication* (2008)

[16] Xen® hypervisor homepage, `http://www.xen.org/`

[17] Weckemann, K. et al: Lessons from a Minimal Middleware for IP-based In-car Communication. In *Proc. of the IEEE IV'12*, pp 686–691, IEEE (2012)

[18] BMW web site. Navigation system Professional, `www.bmw.com/com/en/insights/technology/technology_guide/articles/navigation_system.html`