Effective and Efficient Security Policy Engines for Automotive On-Board Networks

Muhammad Sabir Idrees and Yves Roudier

EURECOM
{muhammad-sabir.idrees,yves.roudier}@eurecom.fr

Abstract. *The configuration of security mechanisms in automotive on-board networks makes it necessary to define and deploy adapted security policies. This paper discusses how to design policy engines that implement an effective enforcement in such architectures despite the complexity of the protocol stacks of on-board electronic control units. It also evaluates how policies expressed in XACML can be adapted to the automotive environment efficiency requirements despite the limited computational power of those units and network bandwidth limitations.*

Keywords: Security Policy, XACML, ASN.1, On-Board Policy Engine

1 Introduction

The changing requirements of automation have triggered a tremendous evolution in automotive on-board systems, making vehicular communication a central feature of modern vehicles. Vehicular on-board networks have long been designed with safety rather than security concerns. The security of bus systems was, to a great extent, achieved through obscurity. Recognized security solutions must now be used in order to mitigate the attacks that recent studies on those architectures have shown to be possible [16] or [25], [14], [33]. Secure communication is a basic need in such architectures and should at least ensure some form of authentication to participants.

Research we conducted in the past 3 years in the European funded project EVITA [21] has been specifically addressing those requirements and appropriate solutions for vehicular on-board networks. This project features a comprehensive security architecture solution covering all aspects of on-board communication (data protection, secure communication, secure and tamper proof execution platform for applications). The security mechanisms that we have defined in this project are in particular coordinated by a security policy aiming at both networking and cryptographic material security. Due to the extended lifetime of vehicles and the need to service them and therefore replace ECUs, it is mandatory that this policy might be updated. We focus in this paper on the networking security policy and discuss how it can be implemented efficiently.
We have approached this problem through the combination of cryptographic

protocols and trusted computing, the latter ensuring the binding of the former to a known platform. This is achieved by associating a hardware security module (or HSM) to every ECU, as illustrated in Figure 1. The security policy to be applied in a vehicle is the combination of an invariant policy for the usage control of cryptographic credentials of electronic control units, and a flexible networking security policy. The credential usage control policy is enforced by the HSM [27] and possibly through the virtualization of the ECUs if applications on the same ECU have to be segregated. In contrast, the networking security policy is enforced by all network elements. Those elements first consist in the protocol stacks implemented at the different ECUs: those stacks implement secure protocols for transport, diagnosis, firmware update, key distribution (see for instance [28]), policy distribution, or even system boot; they have to be linked with the keys initially bound to ECUs by the HSM, which provides trusted authentication as the building block for all those protocols. The interconnection gateways between the different communication buses of the vehicle, or the V2X communication units, or even the tools used to diagnose and service vehicles constitute other network elements where policy enforcement can be done, based on trusted authentication or other security mechanisms like traffic filtering or secure logging.

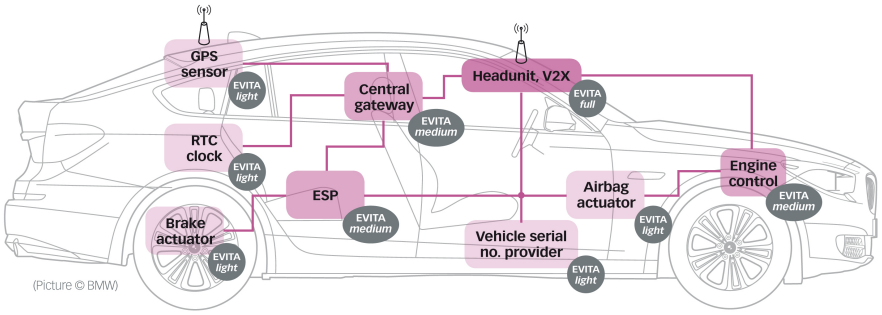


Fig. 1: Automotive on-board network architecture

The remainder of this paper is organized as follows: Section 2 outlines the security policy enforcement architecture, and in what way the security policy supports the flexible deployment and enforcement of the networking security policy. Section 3 discusses how XACML is being used to define the flexible part of such a security policy. Section 4 presents and analyzes performance figures for our policy engine. Section 5 reviews the capabilities of existing access control frameworks in comparison with our own system, with respect to the policy interpretation and enforcement. Finally, the paper summarizes the results that we achieved regarding on-board access control.

2 Security Policy Enforcement Architecture

We first describe the architecture that was adopted in [32] to enforce modular access control. The policy engine, which can be flexibly updated, is deployed within the automotive system environment. The policy that it enforces is defined as a mandatory policy for all communication in the vehicle at a backend system and is transferred to the vehicle. The policy engine is composed of two main components i) A Policy Decision Point (PDP), following the terminology used in XACML, and ii) Policy Enforcement Points (PEPs). Within the scope of EVITA, both components are implemented as a proof of concept within the EMVY middleware framework defined in [6]. EMVY uses a distributed master-client architecture and provides component based templates for introducing security relevant mechanisms for securing communication between ECUs within a vehicle. The PDP is implemented by a component termed the Policy Decision Module (PDM). PEPs are instead implemented at various components of the middleware, as described below. The policy decision module is used as a centralized module accessible from different domains and application as shown in figure 2.

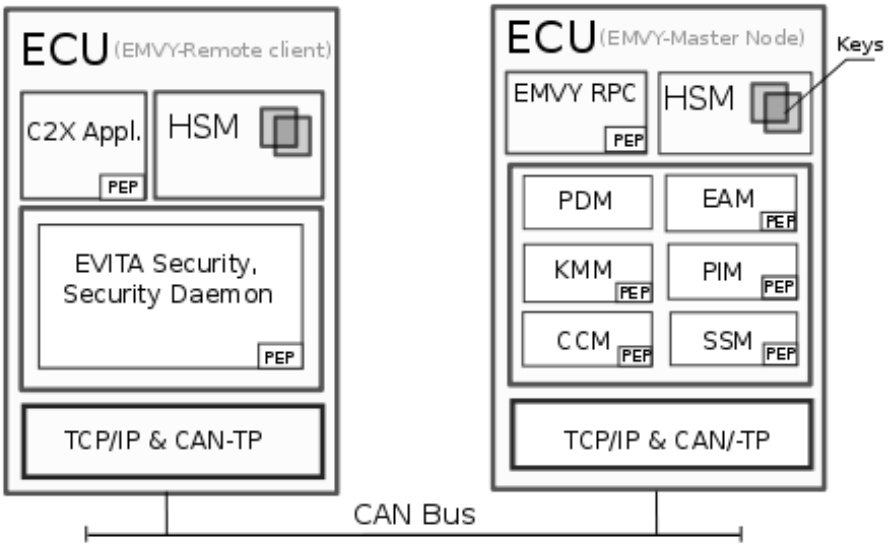


Fig. 2: PDM and PEP deployment

2.1 Policy Enforcement Points

Policy enforcement points (PEP) have to be deployed together with a PDP. PEPs handle access, or communication, or boot sequence validation requests for

instance and have to enforce policy based decisions. Access control in vehicular embedded IT infrastructures has for instance traditionally been developed using a single enforcement point. However, in our fine grained architecture, policy enforcement has to rely on a series of such PEPs at different parts of the communication stacks and middleware, as well as at application level. For instance, deciding on access control for some RPC request requires first allowing network traffic between the communicating parties, then distributing the proper session keys through the key distribution mechanism, then authenticating the communication channel using these keys, then finally making sure that the RPC request can be performed by the entity authenticated at the transport layer. A list of the enforcement points deployed in the car is presented in Table 1.

Security Module	**Module Description**	**Policy Enforcement Point (PEP)**
Communication Control Module (**CCM**)	Handles all communication functionality between EMVY nodes and internal modules.	enforce the filtering policies which act as firewall. Rule enforcement will be based mainly on transport-layer parameters but also on application-layer information.
Entity Authentication Module (**EAM**)	provides authentication, single sign-on/ sign-out security services.	enforce the policy based login and authentication services (e.g., password, smartcard).
Key Management Module (**KMM**)	Distribution of fresh cryptographic key material among clients.	enforce the key generation and group communication security rules.
Platform Integrity Module (**PIM**)	Enhancement of platform security by supplying runtime platform integrity	rule enforcement regarding valid boot integrity measurements.
Secure Storage Module (**SSM**)	Cryptographic storage operations for applications.	enforce the secure storage/access of data security rules (i.e. encrypt the storage device or, can encrypt data objects individually).
Security WatchDog (**SWD**)	Intrusion detection and management services.	policy enforcement rely on a set of rules consists of a attack pattern and an action.

Table 1: Software Security Modules and Policy Enforcement Points (PEPs).

2.2 Handling Policy Decisions

The policy decision point decides based on security policies whether or not access to a particular resource is granted. The decision of the PDP is then enforced by the policy enforcement point (PEP) which drops a message, forwards a message or modifies a message (e.g., encrypts the message with the cryptographic

key of the destination ECU) according to the security policy. A policy decision point usually acts autonomously in its domain where he is assigned and makes decisions in response to an authorization request. By default, a PEP actively queries the PDP for every decision. For instance, an EMVY remote client application (C2X application in figure 2) queries the PDM at the EMVY Master Node for each authorization request. However, a PEP could be pre-configured by the PDM (autonomous PEP). Such an autonomous PEP would act as an ancillary PDP, mostly based on static or cached security decisions, and possibly parameterized decisions depending on security information that are available locally. For instance, during the secure bootstrapping phase, the SSM-PEP acts as an autonomous PEP and decides based on its local policy whether the PDP is allowed to load security policies.

3 Security Policy Expression

As we explained, the onboard network policy has to describe how to configure very different security mechanisms. After exploring several alternatives including drafting our own policy language, we decided on building upon the XACML [17] access control language, obviously for expressing the access control rules of our security policy, such as the definition of secure communication groups and related authorizations at the RPC level, but as well as a more general policy language. XACML provides a flexible and modular way to define and enforce policies, and its decision/enforcement model fits well in distributed environments, even for the on-board embedded system of a vehicle. XACML provides an interchangeable policy format, support for the fine grained description of resources, can describe conditional rights, supports policy combination and conflict resolution. Another important aspect regarding the choice of XACML as our policy language was its independence from a specific implementation and the large number of tools for writing and analyzing any policy. In the case of specific configurations, XACML is flexible enough to represent different security profiles. For instance, the XACML `Policy` element can be used in order to encapsulate complex firewall rules comprising multiple attributes, the source IP address/port numbers being specified in the XACML `Subject` element for instance, and the destination IP address/port number being mapped to the XACML `Resource` element. In a similar way, the XACML `Rule` element can be used to represent distinct firewall rules.
However, due to the embedded nature of the on-board system and its functional and non-functional constraints [18], [15], it is not feasible to transmit, process, or store XML-based policies in the car. For instance, for the ubiquitous CAN bus, which is operated at around 500kbit/s and offers 8 bytes of data payload per packet, verbose formats like XML would constitute a hardly justifiable increase of the bus load. To cope with the above-mentioned limitations, we defined a binary-based security policy language that consumes less bandwidth, is fast to process, and requires less memory (see Section 4 for a performance analysis). We called this representation the Policy decision module Native Language

(PNL). The purpose of the PNL is not to define yet another access control policy language but rather to provide an alternative interchangeable format for XACML policies, that can be used where performance is an issue. We built PNL on ASN.1 standards [31]. These standards are adopted in a wide range of application domains, as in aviation systems for traffic control, mobile networks, network management, secure emails, fast web services, etc., [31]. PNL makes use of these standards, describes a serialized representation of XACML policies in binary format, and ensures that the XACML structure is preserved during serialization. In order to do so, a XACML schema is mapped into a corresponding ASN.1 definition (see listing 1.1). This mapping is based on the ITU-T X.694 standard (Mapping from XML Schemas to ASN.1 modules) [30].

```
XACML DEFINITIONS AUTOMATIC TAGS ::= BEGIN
/* XACML Policy Defintion */
PolicyType ::= SEQUENCE {
  ...
  ruleCombiningAlgId UTF8String,
  target Target,
  choice-list SEQUENCE OF CHOICE {
  ...
  rule Rule
 } OPTIONAL,
 obligations Obligations OPTIONAL
}
Policy ::= PolicyType
/* XACML Target Definition */
TargetType ::= SEQUENCE {
  subjects  Subjects OPTIONAL,
  resources  Resources OPTIONAL,
  actions   Actions OPTIONAL,
  environments  Environments OPTIONAL
}
Target ::= TargetType
...
/* XACML Rule Defintion */
RuleType ::= SEQUENCE {
  effect  EffectType,
  ruleId  UTF8String,
  description  Description OPTIONAL,
  target  Target OPTIONAL,
  condition Condition OPTIONAL
}
Rule ::= Rule
...

END
```

Listing 1.1: Excerpt of a Policy decision module Native Language (PNL) based on ASN.1 Definition. An excerpt of serialized policy using listing 1.1 is presented in Fig.3.

We have implemented the XACML to PNL mapping engine. In our architecture, this encoder resides at the OEM's backend system. The mapping engine is responsible for serializing security policies into the PNL format and then transmitting it to the vehicle. During the serialization process each security policy is verified against a XACML schema. Upon a successful validation, the security policy is serialized into a specific ASN.1 encoding scheme. Due to constraints from other components (e.g., low level drivers, HSM interfaces, etc.) which em-

ploy the ASN.1 DER encoding [29], the security policies are also serialized using DER encoding rules. However, a more efficient binary encoding such as Packed Encoding Rules (PER) can be used if needed. We anticipate that using PER encoding scheme would further enhance performance and latency results.

3.1 Security Policy Configuration

From the on-board PNL configuration perspective, our high-level goal is to auto-configure security policies and appoint access rules from the vehicle ignition stage. This implies that security policy are deserialized during secure bootstrapping phase, while also recognizing an explicit configuration procedures, after secure bootstrapping phase, in order to load new/update policies (i.e., installation of new application with its policy set or update existing policy). Note that the new policy set or security rules which may have an impact on the basic safety of the vehicle are always configured during next vehicle start cycle. Adding/updating safety critical rules while vehicle is running may cause safety critical problems, depending on the security policy responsible for.
In our implementation stack, the bootstrapping protocols are defined to ensure the secure initialization of all security modules and components (see figure 2). On a certain point of the boot strapping procedure, the boot chain send an initialization call to the PDP(s) to load all PNL based security policies (i.e., group communication policy) from its policy database. Note that these security policies are stored in the Secure Storage Module (SSM), which enforces confidentiality, integrity, authenticity, and freshness mechanisms. Thus, require proper authentication and access rights to access these policies. Since during boot strapping process, SSM cannot ask PDP for a decision when PDP is opens/reads from its policy database (which is also stored via SSM), it has to make autonomous decisions. A detail description of autonomous decisions (autonomous PEP) is discussed in section 2.1. On successful verification of access rights, the SSM allow the PDP(s) to read policy set from its policy database. Regarding the integrity of policies, we rely on the security solutions enforced by SSM. However for the policy validation attempt in our prototype implementation, we enforced, that a vehicle will only be started in case of successful configuration of all PDP/autonomous PEP(s) with respective security policies and the vehicle will shutdown momentarily otherwise.

4 Performance Analysis

We have analyzed the performance of the PNL as well as the memory consumption of different policies by varying the number of elements and attributes used. Performance has a special importance with respect to the user experience in automotive environments, where the time to load and configure security policies, and to assess authorization request and response time is a critical issue when the driver waits for his vehicle to start. The deserialization and configuration of authorization/security policies must be performed before receiving any request

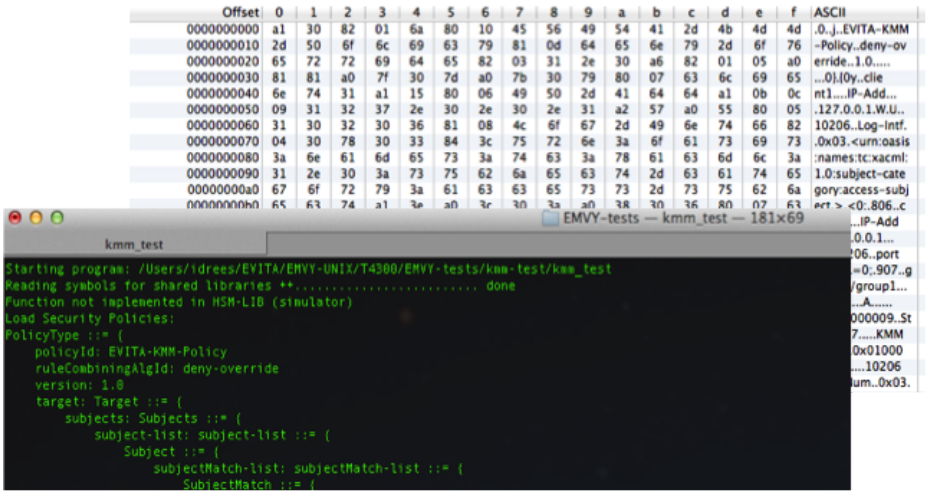


Fig. 3: PDM: On-board policy deserialization and configuration

from a PEP. Hence, the configuration of these security policy is significantly contributing to the overall responsiveness of the policy decision and enforcement at startup time.

4.1 Performance Analysis: Technical Approach

In order to evaluate our results against other XML based access control policies, we are comparing our results with different XML parsers. There have been numerous benchmark studies towards the evaluation of XML parsers [3], [8], [13], [2], [34], [26]. These benchmarks cover several aspects of XML parsing such as performance, schema validation, DOM manipulation, XML security, etc. We used these benchmarks in order to select an agile XML parser implemented in the same language as our own (C/C++), and to run a comparison with our ASN.1 encoded policies. Several lightweight C/C++ XML libraries [3], [2], [1] have been developed for low power devices, with fast parsing capabilities. For instance, the pugixml library enables extremely fast, convenient, and memory-efficient XML document processing. It consists of a DOM-like interface with rich traversal and modification capabilities. However, since pugixml has a DOM parser, it cannot process XML documents that do not fit in memory; also the parser is not a schema validating parser [3], which is a mandatory requirement in our case, for obvious security as well as safety reasons, such as to prevent software compromises.

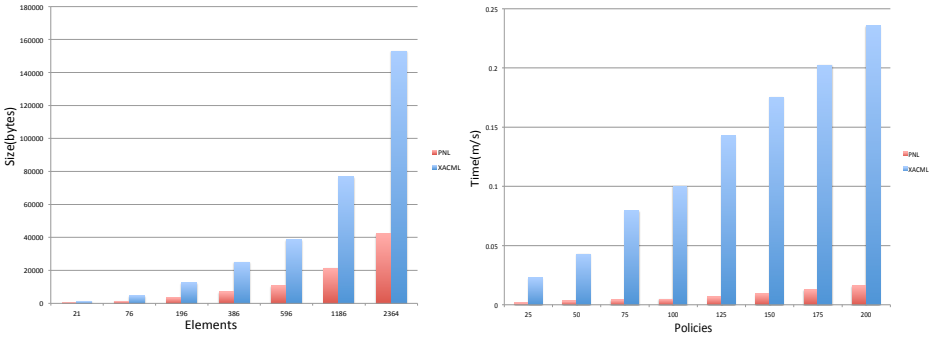
4.2 Experimental Setup and Results

Based on results of these benchmarks and these specific constraints, we decided to evaluate our results with the Gnome XML toolkit (libxml) [4]. According to

the XMLBench Project [8], libxml is the fastest toolkit that has a rich enough set of features. We have generated several XACML policies with various sizes using the UMU XACML editor. The policy tickets are than transformed into a corresponding PNL description, as described in section 3. We have set up a testing environment in order to compare the scalability of our parser with libxml. All presented performance results were obtained using a 64 bit Mac OS X 10.7.2 on a MacBook Pro with 8GB of RAM and a 2.8GHz Intel Core i7 processor. All tests were run in a single user mode without any system services running. We followed the assumptions outlined in [8]: for instance, the time spent to initialize the toolkits is not counted in these results, in order to compare our results with this benchmark. The measurements consist in several latency results which show that the parsing of a PNL encoded policy is lighter and enables a much faster parsing and configuration of security policies. Figure 4.a compares the parsing time with different policy sizes, in which the PNL encoded policy parsing is approximately a third of the XML policy. For 2364 elements (2898 attributes) in a single XACML policy, the PNL encoded policy amounts to 42,368 bytes and the XML encoded one to 152,817 bytes. In the eight tests presented on Figure 4.b, we parsed a single XML policy (21 Elements, 22 Attributes) up to 200 times to understand the scalability. This resulted in parsing times of 0.25ms. In contrast, deserializing using the PNL encoded security policy 200 times takes approximately 0.015ms. Parsing with the PNL encoding is thus about 16 times faster that with an XML based encoding. If we scale up the above presented comparison results for modern ECUs which might only contain 32 bits, 40-MHz to 60 MHz processor [11], the parsing time for 200 PNL policies in an average ECU takes up to 1.05ms. Whereas, the same XML based policies requires approximately 17.5ms. Furthermore, a link to the maximum computational delay to provide safety applications (also related to the maximum number of signature verifications per second) might also highlight the need of such fast and scalable binary based policy language.

5 Related Work

There has been a quite remarkable progress in the area of access control architecture for automotive networks [10], [7], [20], but they appear to have succeeded mostly in terms of requirement specification or have been only concerned with security policies for protecting V2X communications. Gerlach et al. [12] present a C2C communication solution integrating several previous proposals [23], [24], [19] for secure vehicular communications [7]. These proposals consider the car mostly as a single entity, communicating with other cars using secure protocols and thus essentially aim at communication specific security policies enforced at the Communication Unit of the vehicle. Our approach in contrast treats both the expression of V2X and intra-vehicle security policies uniformly. The EASIS project [10] defines a central gateway, a sort of firewall, that is configured so that it denies all data traffic from the external interfaces (e.g. C2C/C2I or Telematics) as a default. Unfortunately, like [7] this proposal is also limited to V2X



(a). Comparison of XML and PNL security policies, with different elements and attributes, in terms of size (bytes) of policy data

(b). Speed up analysis of about 25-200 XML and PNL security policies, where single security policy is constitute of 21 XACML Policy Elements.

Fig. 4: Size of data and increase in speed up factor.

security policy enforcement and not accompanied by any further analysis of the particular requirements/limitation of an in-vehicle architecture with respect to security policies. Zrelli et al. [35] proposed a security framework for the vehicular communication infrastructure implementing access control at both the data link layer and the network layer. However, the proposed solution is solely based on a central policy decision and enforcement module. A single failure in this module may compromise the overall security of the on-board network. Furthermore, this solution is obviously only handling V2X security policy enforcement at the gateway level.

For in-vehicle architectures, numerous authors mention the need for an on-board access control architecture [16], [22], [5]. However, very few solutions have been proposed. A recent security analysis [16] has shown that the risk of attacks on vehicle on-board systems is not anymore of theoretic nature. It depicts several scenarios where access control is either weak or simply not considered, like the firmware update process, which may compromise the overall security of the on-board network, yet no security architecture is described in this work. In [5], a set of cryptographic protocols are discussed to support vehicular use-cases. However, regarding access control, this intra-vehicle security tool box is also limited to the only specification of an API, without any detail about the policy decision engine, practical matters regarding the enforcement architecture, nor implementation perspectives.

Chutorash et al. [9] propose an approach for integrating firewalls in a vehicle communication bus. Firewalls are integrated between application software and between vehicle components. In their approach, filtering rules are applied only on user's request, and commands sent from the HMI, preventing unauthorized access to vehicle components. We see that the practicality of this approach is largely limited by the fact that: `i)` rule enforcement is limited to firewall rules

and more specifically only to user commands. However, in an automotive system, different entities (i.e., security modules as discussed in section 2) are themselves requesters ii) rules are statically defined and remain the same over the vehicle lifetime, and iii) the constraints of embedded vehicular networks regarding notably the policy transmission, processing, or storage and the practical implementation of the proposed approach, are again left out. The OVERSEE Project [22] aims among other objectives at in-vehicle firewall configuration and application level access control using XML based configuration rules. As of now, this project has just begun and no result is available, thus we cannot evaluate the practicality of this approach. However, according to our experience, the performance of verbose formats like XML in a constrained environment has to be closely watched.

6 Conclusion

We have exposed in this paper in what respect the complexity of automotive on-board network architectures and their evolution involve a complex expression of the security policy. Today's automobiles are a perfect example of a system whose security relies on the combination of many different enforcement points in the vehicle on-board network and even in every electronic control unit's communication stack. The role of a security policy in this context is to link the trusted computing base and the trusted credentials it stores with enforcement mechanisms, as well as to connect enforcement mechanisms together. We are taking advantage of the extensibility of XACML subjects to associate attributes, and in particular the means to perform a trusted authentication of electronic control units: this mechanism is at the core of the EVITA approach. The performance of the policy parsing is also very important in a vehicle. We described how the XACML policy can be encoded in ASN.1 in order to make it fit better the resource constrained environment of a vehicle. The policy engine described in this paper was finally deployed in the EVITA project demonstrator - two cars equipped with the EVITA HSMs and software framework - and was successfully used for network filtering, configuring secure group communication, and RPC level access control.

Acknowledgment

This work has been carried out in the EVITA (E-safety Vehicle Intrusion proTected Applications) project, funded by the European Commission within the Seventh Framework Programme for research and technological development.

References

1. Arabica XML and HTML Processing Toolkit. http://www.jezuk.co.uk/cgi-bin/view/arabica.

2. Asm-Xml Benchmark. http://tibleiz.net/asm-xml/benchmark.html.
3. Pugixml Benchmark. http://pugixml.org/benchmark/.
4. The XML C Parser and toolkit of Gnome libxml. http://www.xmlsoft.org.
5. Hagai Bar-El. Intra-Vehicle Information Security Framework. September 2009.
6. BMW. EMVY: The Embedded Vehicular IT Security Construction Kit. Basic Concept, June 2009.
7. C2C-CC. Car2Car Communication Consortium. http://www.car-to-car.org/.
8. S. Chilingaryan. *The XMLBench Project: Comparison of Fast, Multi-platform XML libraries*, pages 21–34. Springer-Verlag, Berlin, Heidelberg, 2009.
9. J. CHUTORASH, Richard. Firewall for vehicle communication bus. In *International Patent Classification 7*, number WO/2000/009363, PCT/US1999/017852. European Patent Office, Feb 2000.
10. EASSIS. Security and firewall concepts for gateways. Technical Report Deliverable D1.2-12, EASIS-Project, 2006.
11. Freescale. Mpc565 reference manual. Technical report, Freescale Semiconductor, 2005.
12. M. Gerlach, A. Festag, T. Leinmüller, G. Goldacker, and C. Harsch. Security architecture for vehicular communication. In *In WIT 2005*, 2005.
13. S. Cheng Haw and G.S.V.R. K. Rao. A comparative study and benchmarking on xml parsers. In *Advanced Communication Technology, The 9th International Conference on*, volume 1, pages 321 –325, feb. 2007.
14. Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Automotive it-security as a challenge: Basic attacks from the black box perspective on the example of privacy threats. In *Computer Safety, Reliability, and Security*, volume 5775 of *Lecture Notes in Computer Science*, pages 145–158. Springer Berlin / Heidelberg, 2009.
15. E. Kelling, M. Friedewald, T. Leimbach, M. Menzel, P. Säger, H. Seudié, and B. Weyl. Specification and evaluation of e-security relevant use cases. Technical Report Deliverable D2.1, EVITA Project, 2009.
16. Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447 –462, May 2010.
17. Tim Moses. eXtensible access control markup language TC v2.0 (XACML), February 2005.
18. N. Navet. Automotive communication systems : from dependability to security. *1st Seminar on Vehicular Communications and Applications (VCA 2011), Luxembourg*, May 2011.
19. P. Papadimitratos. Securing vehicular communications - assumptions, requirements, and principles. In *In Workshop on Embedded Security in Cars (ESCAR)*, 2006.
20. CVIS Project. Cooperative vehicle infrastructure systems. http://www.cvisproject.org/.
21. EVITA Project. E-safety vehicle intrusion protected applications. http://www.evita-project.org.
22. OVESEE Project. Open vehicular secure platform. https://www.oversee-project.com/.
23. M. Raya, P. Papadimitratos, and Jean-Pierre Hubaux. Securing vehicular communications. *IEEE Wireless Communications Magazine*, Vol 13:8–15, 2006.
24. Maxim Raya, Daniel Jungels, Panos Papadimitratos, Imad Aad, and Jean-Pierre Hubaux. Certificate revocation in vehicular networks. Technical report, 2006.

25. Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *Proceedings of the 19th USENIX Security Symposium, Washington DC*, aug 2010.
26. A. Schmidt, F. Waas, M. Kersten, Michael J. Carey, I. Manolescu, and R. Busse. Xmark: A benchmark for xml data management. In *VLDB*, pages 974–985, 2002.
27. H. Schweppe, B. Weyl, Y. Roudier, M. Sabir Idrees, T. Gendrullis, and M. Wolf. Securing car2X applications with effective hardware software codesign for vehicular on-board networks. In *VDI Automotive Security 27. VW-Gemeinschaftstagung Automotive Security, VDI Bericht 2131, Berlin, Germany*, 10 2011.
28. Hendrik Schweppe, Yves Roudier, Benjamin Weyl, Ludovic Apvrille, and Dirk Scheuermann. Car2x communication : securing the last meter - a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography. In *WIVEC 2011, 4th IEEE International Symposium on Wireless Vehicular Communications, San Francisco, CA, United States*, September 2011.
29. International Telecommunication Union. Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690. Technical report, ITU-T, 2002.
30. International Telecommunication Union. Information Technology - ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1, ITU-T Recommendation X.694. Technical report, ITU-T, 2004.
31. International Telecommunication Union. Information Technology - ASN.1 encoding rules: Abstract Syntax Notation one (ASN.1): Specification of basic notation, ITU-T Recommendation X.680. Technical report, ITU-T, 2008.
32. B. Weyl, M. Wolf, F. Zweers, T. Gendrullis, M. Sabir Idrees, Y. Roudier, H. Schweppe, H. Platzdasch, R. E. Khayari, O. Henniger, D. Scheuermann, A. Fuchsa, L. Apvrille, G. Pedroza, H. Seudie, J. Shokrollahi, and A. Keil. Secure On-board Architecture Specification. Technical Report Deliverable D3.2, EVITA Project, 2010.
33. Marko Wolf, Andre Weimerskirch, Christof Paar, and Most Bluetooth. Security in automotive bus systems. In *Proceedings of the Workshop on Embedded Security in Cars (escar) 04*, 2004.
34. Y. Wu, Q. Zhang, Z. Yu, and J. Li. A hybrid parallel processing for xml parsing and schema validation. In *Proceedings of Balisage: The Markup Conference 2008. Balisage Series on Markup Technologies Montréal, Canada*, volume 1, August 12 - 15 2008.
35. S. Zrelli, A. Miyaji, Y. Shinoda, and T. Ernst. Security and access control for vehicular communications. In *Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication*, pages 561–566, Washington, DC, USA, 2008. IEEE Computer Society.