HAL
archives-ouvertes.fr

Expert Knowledge Based Design and Verification of Secure Systems with Embedded Devices

Vasily Desnitsky, Igor Kotenko

▶ **To cite this version:**

Vasily Desnitsky, Igor Kotenko. Expert Knowledge Based Design and Verification of Secure Systems with Embedded Devices. Stephanie Teufel; Tjoa A Min; Ilsun You; Edgar Weippl. International Cross-Domain Conference and Workshop on Availability, Reliability, and Security (CD-ARES), Sep 2014, Fribourg, Switzerland. Springer, Lecture Notes in Computer Science, LNCS-8708, pp.194-210, 2014, Availability, Reliability, and Security in Information Systems. <10.1007/978-3-319-10975-6_15>. <hal-01403995>

HAL Id: hal-01403995
https://hal.inria.fr/hal-01403995

Submitted on 28 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Expert Knowledge based Design and Verification of Secure Systems with Embedded Devices

Vasily Desnitsky[1] and Igor Kotenko[1,2]

[1] Laboratory of Computer Security Problems
St. Petersburg Institute for Informatics and Automation (SPIIRAS)
39, 14 Linija, St. Petersburg, Russia
{desnitsky, ivkote}@comsec.spb.ru
[2] St. Petersburg National Research University of Information Technologies, Mechanics and Optics, 49, Kronverkskiy prospekt, Saint-Petersburg, Russia

Abstract — The sweeping growth of the amount of embedded devices together with their extensive spread pose extensively new design challenges for protection of embedded systems against a wide set of security threats. The embedded device specificity implies combined protection mechanisms require effective resource consumption of their software/hardware modules. At that the design complexity of modern embedded devices, characterized by the proper security level and acceptable resource consumption, is determined by a low structuring and formalization of security knowledge. The paper proposes an approach to elicit security knowledge for subsequent use in automated design and verification tools for secure systems with embedded devices.

Keywords: embedded security, embedded device design and verification, security components, expert knowledge.

1 Introduction

The sweeping growth of the amount of embedded devices and their extensive spread pose extensively the problem of design of their protection mechanisms against a wide range of information security threats. Mostly design complexity of secure embedded devices is determined by a low structuring and formalization of the embedded security knowledge. The specificity of the field is appearance of new expert knowledge, their obsolescence, information acquisition from various sources, such as embedded device industry, research and analytical works in information security and software engineering, experience in exploitation of existing information and telecommunication systems, through security and trust analysis of systems.

Embedded device specificity, leading to the need of specific approaches to their design and analysis, includes highly specialized purpose of devices and hence the domain specific character of their protection, considerable constraints on volumes of hardware resources, specific sets of vulnerabilities and possible attacks to compromise embedded device and its services, multi-component based approach [21] and therefore

possible implicit connections and hidden conflicts between security components arising from the absence of their a priori joint conformity.

As a result, to solve security issues to the full extent information security experts of high qualification are required to be involved in the course of all stages of the design process. In general the search and involvement of such experts complicates the design process significantly, introducing new iterations, feedbacks between the developers, experts and other roles involved as well as increases financial costs to accomplish the development process. At that the current trend in the field of embedded device development is to delegate some part of expert duties to developers, owing to the application of specialized automated techniques and software tools for design, verification, testing, evaluation and implementation of embedded devices. That is knowledge on particular industry systems with embedded devices along with expert knowledge are subjected to generalization and transformed into particular techniques and tools for subsequent application by devices developers.

Another trend in embedded device development is to produce families of devices with a basic functionalities and different extra details determining peculiarities of the device exploitation and finally the cost of the device. As a result there is no necessity to fulfill the expert assisted design process fully for each device within a given family. Instead one should conduct some adaptation of already developed protection procedures and design protection procedures, taking into account the specificity of particular devices that mostly can be delegated to the developer.

The main goal is to form, structure and refine expert knowledge characterizing various design and verification aspects of embedded security mechanisms as well as to search new ones and adopt existing techniques and automated software tools for their subsequent use by developers with embedded devices. The main contribution of the paper is a proposed technique for design and verification on the base of the revealed expert knowledge. The technique is targeted on development of combined security mechanisms for embedded devices, considering resource consumption metrics, possible conflicts and anomalies of security components and information flows. The technique is characterized by engaged specific expert information on hardware resources, typical conflicts and anomalies.

Systems with embedded devices are getting spread in the sphere of home land defense and security. Such systems allow arrangement of collaborative coherent and secure operation of heterogeneous embedded and mobile devices, sensors, servers and other devices as well as various services and agents engaged. Development of security mechanisms for these systems taking into account specificity of particular devices and expert knowledge in the field will facilitate both global and national defense capability.

The paper represents a logical continuation of our published papers on design and analysis of secure systems with embedded devices [8, 6, 28]. Particularly, in the paper we (1) propose a heuristic to determine an order of consideration of hardware resources of the configurable device, depending on its functional and non-functional features, (2) extend the list of typical conflicts between security components of embedded devices, (3) reveal relevant types of information flow anomalies and the ways of their detection with reference to information flow analysis inside systems with embedded devices, (4) present the developed software prototype for detection of information flow anomalies, (5) fulfill an analysis of the proposed approaches. The rest of the paper is organized as follows. In section 2 the related works are surveyed. Section 3

encompasses the basic elements of the proposed technique, including configuring security components, detection of hidden conflicts, and verification of network information flows. Section 4 comprises the domain specific analysis of the field of embedded security. It outlines a fragment of the case study used as expert knowledge sources for the proposed technique. Section 5 exposes the revealed expert knowledge used in configuration and verification processes. Issues of software implementation and discussion are presented in section 6.

2 Related works

A multi-component based approach to design systems with embedded devices got a relatively wide application [21] particularly within mobile operating systems Android. The protection system is represented as a set of interacting software and software/hardware components, each of them being in charge of particular functional security requirements. At that the process for combining security components, taking into account their peculiarities into a single mechanism is configuration of security components [8]. The drawbacks of the approach are possible implicit connections and hidden conflicts between security components arising from the absence of their a priori joint conformity. In [3, 17, 25, 13, 19] the core problems in the field of embedded device security are presented as particular security domain problems such as user identification, local secure data storage, software resistance to modifications, secure access, side channel attacks protection and others. Contemporary security mechanisms of embedded devices mostly are oriented to particular specific vulnerabilities. In [1, 18, 25, 28] various classifications of vulnerabilities, embedded device intruders are proposed, exposing intruder capabilities, competence and access type. At that combining various heterogeneous protection means within a single device, interrelations between them and issues of their integration correctness are not presented in existing works to the full extent.

The importance of the embedded device development, taking into account acceptable energy and computational expenses along with higher security level are uncovered in [9, 16, 27]. Besides granting necessary hardware and energy resources to the device and its services, a special issue is DoS attacks targeted on exhaustion of device energy resources [22, 33]. At that this kind of attacks is not detected by conventional antivirus solutions and other ones, but aimlessly waste energy resources through the use of the most energy expensive hardware components like Wi-Fi and Bluetooth modules or screens, complicating the further functioning of the device. Therefore a complex security mechanism should contain software and hardware modules against various relevant security vulnerabilities, taking into account possible implicit connections and inconsistencies between particular protection modules.

As a way to achieve a tradeoff between the security of the device and its resource consumption, Gogniat et al. [12] propose the usage of reconfigurable security primitives on the base of dynamic adaptation of the device architecture, depending on a state of the device and its environment. The adaptation suggested in [12] is based on, first, dynamic switching between a number of mechanisms integrated in the device and, second, update of these mechanisms.

Configuration processes along with analysis of hardware resource constraints and time expenses are of importance for development of end-products [15, 34, 35]. At that configuring facilitates a shift from development of a mass product to a customized one adjusted to the needs of a particular client [30].

As design case tools the specific UML profiles are used in the industry, holding relevant embedded security peculiarities, particular requirements, vulnerabilities, security components and their properties and connections between them. In particular in [28, 29, 31] Domain Specific Models are introduced to model and analyze security mechanisms for systems with embedded devices. In essence each domain is oriented to representation of the device or the whole system in the context of some particular security feature, such as secure storage domain, secure communication domain, user authentication one, etc. An advantage of the approach is delimitation of the design process tasks, responsibilities and roles involved as well as the use of expert knowledge in embedded security field to produce a device protection system. Software tool SPT (SecFutur Process Tool) [31] implementing the concept of domain specific models represents an extension to general purpose design environment MagicDraw. Model-driven design and analysis of embedded devices and real time systems are presented in MARTE framework [21] defining a complex UML based conception of software and hardware qualities of a device to support its specification, synthesis, verification, validation, performance evaluation, quantity analysis and device certification with the use of UML profiles. However UML based software tools for design and verification are oriented to development of static structure of devices, their specification and subsequent software/hardware implementation without evaluation of dynamically changing characteristics such as resource consumption laying beyond the scope of conventional UML apparatus.

3 Design and verification approaches

This section presents the basic elements of the proposed technique for design and verification of systems with embedded devices. The technique includes the following stages: (1) configuring security components of an embedded device; (2) verification of its protection system to reveal hidden conflicts; (3) verification of network information flows. The essence of the technique is in the use of specific heuristic based embedded security related knowledge as completed design and verification patterns along with the use of methods of model checking, discrete optimization and decision making.

A. Configuring security components

Correlation of security level of embedded devices and their various non-functional characteristics such as resource consumption represents a challenge in the field of secure embedded device development. Often the absence of effective design-time tools to develop combined security mechanisms complicates or even makes virtually impracticable the implementation of sound protection system. The proposed approach to design the protection systems for embedded devices is realized in accordance with multi-component based approach, taking into consideration both functional and non-functional requirements and limitations of the device and security components as well as resource consumption criteria to obtain the most effective solutions customized by

non-functional constraints of a particular kind of devices. In essence a resource consumption criterion determines a sequence of hardware resources ordered by their criticality level. At that a discrete optimization problem is formed on a set of security configurations, while its solution allows getting the optimal configuration [8].

The goal of the proposed approach is to determine the most resource effective (optimal) configuration of the protection system on the base of input data on the device and its security components. The configuration is intended to be integrated into the protection system of the device. Finding the optimal configuration will allow ultimately improvement of device protection effectiveness. The choice of an optimal configuration depends on the following factors: (1) device hardware capabilities and volumes of resources to be allotted to support the protection system; (2) needs of the resources for particular security components. For instance asymmetric encryption as a rule requires significant computational expenses; a remote attestation component requires additional network bandwidth expenses leading to higher consumption of energy resources; (3) device peculiarities, scenario of the device, its autonomy, mobility as well as other characteristics and requirements to the device and protection.

Configuring is conducted in automated mode on the base of developed decision making tool to choose optimal configurations. At that, resource consumption criteria are set manually, depending on non-security requirements, peculiarities of the device and its protection system under configuration process. Therefore we propose to use a specific heuristic to determine an order of consideration of hardware resources in the configuration process, depending on functional and non-functional features of the device. Further in sections 4 and 5 in framework of a domain-specific analysis of the field of embedded security we survey shortly a case study as well as a heuristic based on the analysis of this case study. In essence configuring represents a discrete multi-criteria optimization problem on the set of security components. Due to the finite and relatively small amount of security component alternatives available in the design process there is no need to look for or create any specific methods to solve the optimization problem in a short period of time. In fact the proposed heuristic should be used by the device developer to form particular optimization problem constraints and their order properly.

B. Detection of hidden conflicts between security components

Multi-component based approach to design of embedded devices and in particular their protection systems cause a problem of correct and secure combined use of several security components. Even assuming individually each security component has no internal inconsistencies and vulnerabilities, the combined protection mechanism nonetheless can be subject to hidden conflicts of different character.

Such conflicts may lead to security vulnerabilities in the protection system, incorrect work of the protection system and even business functions of the device. The main complexity is that these conflicts may appear at the exploitation stage of the device only. Therefore their elimination may require lots of financial costs and industrial expenses. Thus an important task is seen to detect known kinds of hidden conflicts between security conflicts in design-time. In section 4 we present a number of typical conflicts as a piece of expert knowledge and their examples. These conflicts were got heuristically through an analysis of existing systems with embedded devices and a range of papers on embedded security [5, 31].

C. *Verification of network information flows*

The goal of network information flow verification is to evaluate security level of the developed information system with embedded devices. The verification is conducted through checking correctness of the security policy for the system and determine to what extent information flows in the real system correspond to the policy.

By information flow we mean summation of information passed between two or more interacting objects. Information flow security policy represents a set of rules determining which information flows in the system are permitted or prohibited.

Conventionally information flow analysis is conducted at three levels, (1) *hardware* – as an analysis of ties between the microcircuits [4], (2) *software* – as an analysis of the source code running on the device [24], and (3) *network* – as analysis of network connections in systems with embedded devices. Information flow analysis at these levels are covered in detail in existing literature [24, 14]. Amount of papers on verification of network information flows is significantly less than ones on software and hardware flows. The concept of information flow is widely used in security evaluation of a route and network effectiveness evaluation [2, 32]. Although these studies are not directly related to the types of data transmitted by information flows in the network, but nonetheless, they can be used for modeling information flows. Usually information flows between nodes are specified as a directed acyclic graph. Thus, to reveal covert channels the topological analysis described in [26] can be applied to this graph. In this paper we apply model checking to verify security policy rules for information flows. In general, checking correctness of network information flows is an integral part of the design process. Carrying out such verification at the initial design stages provides early detection of contradictions in the security policy and inconsistencies of the information system topology.

Verification of network information flows at the initial design stages represents the static analysis of a system. In contrast to the dynamic analysis including testing of end devices on the basis of attack vectors the proposed verification approach can reduce the number and complexity of actions that need to be repeated after the design errors become fixed. In general, the static approach is to analyze the structure of the information system and its characteristics as system models at different levels of abstraction [20] (security policies and business logic). To verify the security policy rules regarding checking network information flows we propose to apply model checking, using SPIN tool and PROMELA language. Checking information flows is carried out on a model of the system, since information flow verification on the real network would be much more difficult due to the need to involve specialized equipment, software tools and staff of qualification. Enumerating the policy rules is realized in order of decreasing priority until some rule holds. Prioritization allows organization of more complex management of interrelated policy rules.

4 Analysis of expert knowledge sources

We used three industrial systems with embedded devices (case study) as a source of expert knowledge in the field of embedded systems [31], namely a system of remote automated control of energy consumption by consumers (*abr. MD*), a quickly deployable emergency management system (*abr. TMN*) and a system providing consumers with digital media services (*abr. STB*). The choice of these three case

studies is determined by their different structure, purposes, functional and security features. The expert knowledge obtained through the analysis of these systems can be generalized and used as a completed design and verification patterns in the development of new systems.

The following patterns, forming the proposed technique, are related to expert knowledge. These are particular security requirements in the shape of functional protection properties and possible alternatives for choosing security components; information on non-security features and internal ties of both an embedded device and its security system to be the base of resource consumption construction; possible types of conflicts that security components are involved in; possible types of information flow anomalies and ways of their detection.

A brief description of the system *MD* developed by Mixed-mode [31] is presented below. The system represents a network containing digital trusted electricity meters on the client side, a trusted server and database as well as an infrastructure for communications between devices and their management. The system is characterized by a branched network topology, the presence of technical personnel roles in charge of installation, gauging and support of the system devices as well as a need to protect the devices and software services from malicious users and third parties trying to compromise the system. The system contains trusted sensor modules (TSM) to measure electricity of households (Fig. 1).

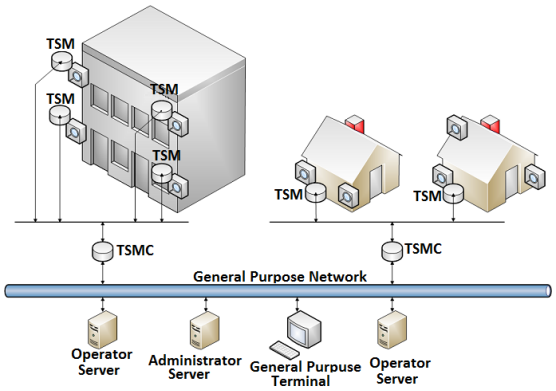

Fig. 1. The system of remote automated control of energy consumption [31]

Measurement data got from each TSM are sent, using the local data bus to a trusted sensor module collector (TSMC). For remote access and control of TSM and TSMC a general-purpose terminal belonging to a general purpose network is used. TSM and TSMC are considered as functional physical modules not necessarily standing alone. However they can be implemented within a single device [31].

On the base of the analysis of the system specification and models of embedded device intruders the developers have provided the following functional protection properties, each of them being associated with some security component [31]. These are the integrity of data transmitted to and from the device, in particular, the target data of the current energy consumption on the client side; the integrity of data stored locally on the device; the confidentiality of data transmitted to and from the device; confidentiality of data stored locally on the device; data flow control in accordance

with a given security policy; monitoring unauthorized and potentially dangerous action in the system; implementation of protection against unexpected data; presence of data protection from destruction and loss during their transmission or processing due to software failures; presence of a mechanism for safe update of security features; ability to identify compromised and alien devices and components; presence of a mechanism for detection of anomalies in the measured data received from the device; realization of local role based access to the device; continuous integrity monitoring software components of the device.

Non-security TSM related expert knowledge items are as follows: presence of a permanent power source; TSM does not store large amounts of data (only stores the measurement data), data loss is not critical; no complicated calculations (since the main function is reading and transfer of data from the sensor); requirement of timeliness of the business process function; importance of communication services, volumes of business data (measurement data) in the device are small.

The search of typical conflicts and anomalies in the system represents a heuristic analysis of specifications and system models, taking into account already known types of conflicts and anomalies listed in section 5. Specifically for the system *MD* the policy rules constituting a shadowing anomaly have been analyzed.

5 Expert Knowledge

A. Configuring security components

An optimal configuration choice is carried out using lexicographic ordering of specified resource consumption criteria. The ordering is based on a heuristic to determine the order of consideration of hardware resources in the configuration process, depending on the functional and non-functional features of the configurable device. The heuristic is based on expert knowledge derived from the analysis of three industrial systems with embedded devices (system *MD, TMN, STB*).

The heuristic represents a general algorithm for prioritization of hardware resources of an embedded device. A set of signs of embedded devices and the services they provide, having the influence on resource consumption is specified. We introduced a three-point ranking for resources according to their criticality to execution of the target device functions (*0 means the resource is noncritical, 1 means low criticality and 2 means high criticality*). By experts a rank value is specified for each sign of the core device of each of the three systems in use. Table 1 shows the four types of hardware resources in accordance with the methodology MARTE [21], a set of signs for each of them, references to the analyzed systems that have devices with the regarded signs and the corresponding ranks. Thus, the ranks obtained on the basis of expert evaluation of the analyzed systems are taken as ranks of the signs themselves. Hence these rank values can be used for express ranking of resources of the device by its developer without additional participation of experts.

Thus, in configuring process the device characteristic signs are identified from the list of available ones. After that each resource is assigned a maximum value of rank over all held signs corresponding to a given resource. As a result, the considered hardware resources are ordered according to decreasing their ranks. If two or more

resources have the same rank value, the default order *<HW_PowerSupply, HW_StorageManager, HW_Computing, HW_Communication>* is used. It was defined BY experts as a priori and the typical for the most existing systems. It is assumed if necessary this heuristic may be refined by adding additional signs, resources, analyzed systems and devices to consider as expert knowledge.

Table 1. A heuristic for choosing resource consumption criteria

Resource type according to MARTE	Signs of embedded devices and its services	Abbreviation of the systems with devices of the sign	Rank
HW_PowerSupply (energy consumption resource)	The presence of a permanent power source	*MD, STB*	0
	Possibility of replacing the device or battery without damage to the provided services	*TMN*	1
	Sporadic access to a centralized power supply	*TMN*	1
	High dependency of the mission goal achievement on energy resources	*TMN*	2
HW_StorageManager (storage resource)	The device does not store large amounts of data, loss of data is not critical	*MD*	0
	Storing large amounts of data, loss of data is not critical	*STB*	1
	Storing large or unlimited amounts of data, the loss is critical	*TMN*	2
HW_Computing (computational resource)	No complex calculations, no requirements of message delivery timeliness	–	0
	No complex calculations, major timeliness	*MD*	1
	Complex calculations, minor timeliness	*STB*	2
	Complex calculations, major timeliness	*TMN*	2
HW_Communication (communicational resource)	No communications (or they are not obligatory for the device services)	–	0
	Importance of communications for the device services, minor data volumes	*MD*	1
	Importance of communications, large data	*STB, TMN*	2

B. Hidden conflicts of security components

Analysis of hidden conflicts of security components is an integral part of the effective configuration selection process and is performed by embedded device developer during the protection system design. In essence this is a heuristic analysis aimed at identification of known kinds of hidden conflicts, which the security components of embedded devices are involved in [8].

Generally a conflict is regarded as a relationship between two or more security components and represents a contradiction between the functional of several security components, any their non-functional limitations and/or software/hardware platform of the device. The peculiarity of such conflicts is that as a rule they become apparent under certain conditions only and. Therefore, it is difficult to detect them during testing end devices by the use of attack vectors. Early design-time detection of conflicts in the process of integrating security components will help to reduce the number of iterations of the device development process. Besides for a conflict to be appeared not only the fact of integration of multiple security components with specified security functional is important, the way of their integration is significant as well. Specifically two components with opposite protection features can be in a conflict if they are performed simultaneously and interact within a common hardware/software context, for example, they use share data structure, memory, file, communication channel and so on. Knowledge of known types of conflicts is produced by expert analysis, modeling and development of new information systems with embedded devices. It seems appropriate

to keep a list of previously discovered types of conflicts, regarding domain-specific nature of each particular system. As a consequence as the specification of the combined device protection system as well as specifications of considered security components should be analyzed together by the developers for the presence of conflicts from the list. Differences between the nature of each particular conflict, amounts of the involved security components and their protection functional, peculiarities of the components interactions and their integration as well as domain-specific character cause the development of comprehensive classification covering all possible hidden conflicts seems infeasible at the moment. However, in the design process a particular classification of conflicts (e.g. according to the type of the involved objects) can be used as an expert knowledge by the device developer of the protection system to realize a directional search of possible conflicts (Table 2).

Table 2. Types of conflicts between security components

Type 1 – conflict due to a lack of consistency between a security component and the device specification
Type 2 – conflict between the protection functions of several security components
Type 3 – conflict between several basic components within a complex security component

Fig. 2 schematically shows the three types of conflict discussed. Examples of each of the three above conflict types are presented in Table 3.

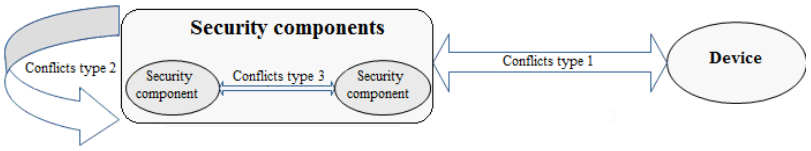

Fig. 2. Three types of security component conflicts

Resolving such conflicts is individual and determined by the specificity of a particular conflict and its security components involved. As resolution options a revision of one or several security components, changing the way their integration or correcting security requirements can be considered.

Table 3. Examples of expert knowledge on conflicts

Conflict type	***Conflict example***
Type 1	*Security_component* = "TPM based secure module for storing confidential customer data"; *Safity_requirement* = "to double customer data by an extra hardware storage module"; *Conflict* = "assuming the only TPM in the device the unprotected doubling violates data confidentiality"
Type 2	*Security_component_1* = "backup component for critical customer data"; *Security_component_2* = "component for secure guaranteed deletion of critical customer data after some specific event happens"; *Conflict* = "inconsistent application of the both components to the same data causes a conflict due to a logical opposite of the their security features"
Type 3	*Security_requirement* = "to implement RAID based redundant and high-performance storage of business data by two (or more) secure hardware units"; *Assumption* = "the inconsistent parameters of the units (e.g. different capacity of the units or their writing speeds)"; *Conflict* = "the units are correct themselves, but they do not implement RAID"

C. Network information flow verification

For verification of network information flows the expert knowledge includes instances of security policy anomalies and methods for their detection. Consider one type of anomalies more in detail, “shadowing” anomaly. The presence of this anomaly supposed that a rule never works because there are one or more rules with higher priorities "overlapping" it. This anomaly indicates a probable error in the policy, which should be reviewed.

Network information flows and policy rules are specified by the following tuples:

$$InformationFlow = < host1, host2, user1, user2, interface1, interface2, type >,$$

$$FilteringRule = < host1, host2, user1, user2, interface1, interface2, type, action >,$$

where *host1*, *host2* – sending and receiving hosts, respectively; *user1*, *user2* – user sending and receiving user; *interface1*, *interface2* – types of hardware Interfaces of the sender and recipient; *type* – type of the information flow.

Type of information flow refers to a kind of data that the flow encapsulates. Information flow types by both the kind of transmitted information (e.g., user data, critical data, checksums, encryption keys, security certificate, etc.) and the format which the information is presented in (e.g., unencrypted and encrypted messages, compressed message).

The essence of model checking, applying to anomaly detection consists in iterating states the system can move into, depending on the emerging information flows and responses from the component making decisions on policy based permission or rejection of such requests. When iterating the sequence of actions depends on conditions formulated in a language of linear temporal logic and express correct states of the system [6, 20]. State of the system is determined by a set of variables and state change is caused by concurrent processes running in the system. A process to be executed in the next time is chosen randomly. The system considers all the possible sequences of steps for specific processes and signals potentially incorrect state. After that, the user is given a track, i.e. a sequence of steps leading to an incorrect state of the system with respect to given conditions. Basic input of verification of network information flows includes, first, descriptions of policy rules and, second, the structure of the network in the system description language and detectable types of anomalies.

At the first stage of verification input data is converted into an internal format of the verification system. Then, at a second stage, a general model of the system is built to verify prohibiting and permitting rules for information flows. The model is presented in the form of a finite state machine and initialized by the input data in internal format. In the model the anomalies are expressed by formal statements. According to model checking paradigm these formal statements represent properties of correctness, which violation brings the analyzed system in an incorrect state. At the third stage the general model is verified by a model checker tool. In the verification process all incorrect state of the system are revealed. At the final verification stage the obtained are subjected to interpretation. If any anomaly instances are detected, it is created a description containing situation and the information flow leading to the appearance of the anomaly and its type [20].

For the case of a shading anomaly the verification includes: (1) generating a set of testing flows (the flows are formed on the basis of the so called “boundary” values of the policy rules, i.e. the flows are constructed through any possible combinations of parameters taken from the rule statements); (2) sequential application of the policy to

each information flow; thus each time the rule holds it is marked as *held*; (3) search on the set of rules to identify rules did not hold even once.

Therefore, verification allows getting a set of results, each of them being a pair $<A, (B_1,...B_n)>$, where A represents an anomalous rule, $B_1,...B_n$ are higher priority rules shadowing it. $B_1,...B_n$ are isolated by an extra pass of the policy by running those testing flows that meet the conditions of rule A.

6 Software Implementation and Discussion

A software prototype developed is used within the proposed technique of design and verification of systems with embedded devices. The prototype includes a design-time means for making decisions on choosing optimal configurations and for verification of network information flows.

The architecture of the tool for making decisions on choosing optimal configurations on the basis of UML class diagrams is presented in Fig. 3. At the architecture there are its grouped elements in charge of the protected device and its properties; security components; classifications of properties of the device and its individual security components; optimality criteria as well as configuration function of and check of configuration admissibility. This tool includes the following main features: (1) configuration function, that forms an optimal configuration according to the given constraints and a list of security components (function *configure*); (2) check function for verification of configuration admissibility (function *verify*).

As practice shows, often in the development of combined protection systems with embedded devices the choice of security components is realized by developers intuitively without any experimental evaluations on an already produced device with integrated protection and without taking into account any design-time system models and heuristics. Experiments on modeling strategy for choosing pseudo optimal sets of security components on the base of greedy algorithms have been realized. The strategy represents a procedure for a sequentially organized choice and refinement of security components of the sought configuration iteratively for each security requirement. In fact, this procedure works successively, for each functional protection property choosing a security component from the available ones that consumes the least amount of hardware resources according to their order determined by the heuristic. The averaged experimental data allow us to deduce that the proposed configuration process results in more effective solutions of combined protection. At that the combined protection effectiveness is meant as achievement of minimal resource consumption of a set of security components providing the given security features. More detailed description of the configuration tool, its performance evaluation as well as fragments of the GUI are given in [8].

The proposed verification of network information flows has been implemented for the analysis of security level of the system of automated control of energy consumption by end customers (system *MD*) with the following limitations. Due to technical simplifications, a limitation of the implementation is setting parameters of the policy both by defining concrete values or rules (specific hosts, interfaces, users) and by using special identifiers *any*, defining all possible values for a parameter. Generally it is assumed setting undefined sets of parameters of rules (in particular as the use of

structures such as *"all the values, excepting x1, x2, x3"*). The policy rules for the *MD* case study were established based on available system specifications.

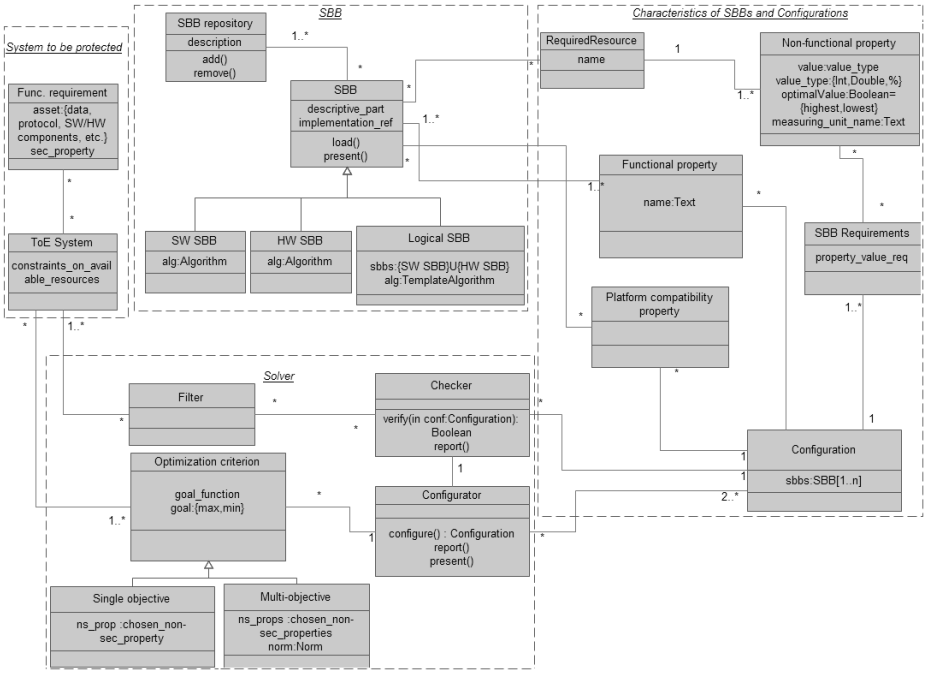

Fig. 3. A tool for configuring security components

We performed experiments, introducing shadowing anomaly instances into the policy. These anomalies simulate potential errors in the process of the policy development. During the technique all the instances were revealed. After verification completed the original policy was subjected to corrections, the verification repeated and new policy admitted as free of shadowing anomalies.

A piece of the requirements got from *MD* case study specification presented below.

> *"The privacy non-relevant data is generated by and temporarily stored on the trusted meter. It is displayed on the trusted meters local display.*
>
> - *Any user shall be able to read the privacy non-relevant data by using the local interface of the Trusted Meter, and only by using the local interface."*

Fig. 4 shows a fragment of a security policy, two policy rules in PROMELA language specifying the given requirement and presenting a shadowing anomaly (rule 0 is presented in lines 82-92, whereas rule 1 is in lines 94-104). In accordance with the location the rule 0 has a higher priority than the rule 1. This anomaly is the result of an incorrect indication of the values of interfaces of the source device (interface1).

Fig. 5 shows a window with a trace from the use of SPIN. In particular, it is shown that the rule 1 is indicated as abnormal.

```
82      rule0.user1 = any_user;
83      rule0.user2 = any_user;
84      rule0.interface1 = any_interface;
85      rule0.interface2 = any_interface;
86      rule0.host1 = TM;
87      rule0.host2 = any_host;
88      rule0.type = Privacy_non_relevant_data;
89      rule0.action = allow;
90      rule0.isHeld = false;
91      rule0.id = 0;
92      storage.policyRules!rule0;
93
94      rule1.user1 = any_user;
95      rule1.user2 = any_user;
96      rule1.interface1 = local_interface;
97      rule1.interface2 = any_interface;
98      rule1.host1 = TM;
99      rule1.host2 = any_host;
100     rule1.type = Privacy_non_relevant_data;
101     rule1.action = deny;
102     rule1.isHeld = false;
103     rule1.id = 1;
104     storage.policyRules!rule1;
```

Fig. 4. Example of rules containing a shadowing anomaly

```
  i=1
517:    proc  3 (printResults) IF0a.pml:790 (state 10) [printf('\\n    i=%d\\n',i)]
Considering rule #1
518:    proc  3 (printResults) IF0a.pml:791 (state 11) [printf('\\n\\n Considering rule #%d\\n\\n',rul
e.id)]
spin: IF0a.pml:792, Error: assertion violated
spin: text of failed assertion: assert(rule.isHeld)
#processes: 4
519:    proc  3 (printResults) IF0a.pml:792 (state 12)
519:    proc  2 (generateIFs) IF0a.pml:761 (state 169)
519:    proc  1 (initModel) IF0a.pml:474 (state 25)
519:    proc  0 (:init:) IF0a.pml:822 (state 2)
4 processes created
```

Fig. 5. The technique output

The experiments on modeling a large number of involved objects, roles, data types and permitting/prohibiting rules confirmed the effectiveness of the proposed verification for the design of the automated control system of energy consumption (*MD*). Since the typical conflicts and anomalies are detected mostly heuristically, it is difficult to deduce about any universal ways to resolve them. Elimination of a conflict/anomaly is determined, first of all, by its context including specific security requirements and assumptions, information security risks, modes of the device, involved security components, used interfaces, etc. To verify network control information flows of a security policy it is not sufficient to use pairwise comparisons of the policy rules only. In fact an analysis of the policy rules holdings in dynamic (i.e. model checking) is needed. In general, compared with the classical network architecture, the specificity of information systems with embedded devices in the task of verification of network information flow contains presence of a branched network topology based on heterogeneous embedded devices with different types of communications and types of hardware/software interface being entry and exit points for information flows, and variability of the structure of such systems throughout its work. An advantage of the proposed verification of information flows is to ensure the system security, assuming the same behavior of the model and the real system. Disadvantages include a large amount of computational resources required to analyze complex models; possible false positives, i.e. warnings on anomalies missing the real system; and incompleteness, as instead of the real system its model is verified.

7 Conclusion

The paper focused at the technique for design and verification of information systems with embedded devices. It is oriented at development and comprehensive analysis of the combined security mechanisms to protect embedded devices on the basis of resource consumption metrics, potential conflicts and anomalies between protection components and information flows. The technique is based on domain-specific analysis of several case studies and characterized by specific expert information on hardware resources of embedded devices, typical conflicts and anomalies. The technique peculiarities include the use of specialized heuristic knowledge in the field of embedded security as completed design and verification patterns, applying methods of model checking, discrete optimization and decision-making theory.

As future research we are planning to identify and use additional expert knowledge by analysis of specifications new case studies, research papers, technical and analytical reports in the field. It is expected to expand the list of typical conflicts and anomalies and do the further work on SPIN based verification component. Knowledge identified in the research is planned to be organized in an ontological form, using a modeling environment Protégé. The peculiarity of this representation is unification of expert information for its subsequent use by device developers both in decision-making design directly and as input for automated development tools.

Acknowledgements. This research is being supported by grants of the Russian Foundation of Basic Research (projects 13-01-00843, 13-07-13159, 14-07-00697, 14-07-00417), the Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (2.2), by Government of the Russian Federation, Grant 074-U01, the State contract #14.BBB.21.0097, the project ENGENSEC of the TEMPUS program and the FP7 SecFutur project.

References

1. Abraham, D.G., Dolan, G.M., Double, G.P., and Stevens, J.V.: Transaction security system. In: IBM Systems Journal, 30(2), pp.206–228 (1991)
2. Agaskar, A., He, T., and Tong, L.: Distributed Detection of Multi-hop Information Flows with Fusion Capacity Constraints. In: IEEE Transactions on Signal Processing, Vol.58, No.6, pp.3373–3383 (2010)
3. Arbaugh, W.A., van Doorn, L.: Embedded security: challenges and concerns. In: Computer journal, Vol. 34, No. 10, pp.40–41 (2001)
4. Braghin, C., Sharygina, N., and Barone-Adesi K.: A model checking-based approach for security policy verification of mobile systems. In: Formal Aspects of Computing Journal, pp.627-648 (2011)
5. Burleson, W., Clark, S.S., Ransford, B., and Fu K.: Design challenges for secure implantable medical devices. In: Design Automation Conference (DAC), 49th ACM/EDAC/IEEE, pp.12-17 (2012)
6. Chechulin, A., Kotenko I., and Desnitsky V.: An Approach for Network Information Flow Analysis for Systems of Embedded Components. In: LNCS, Vol. 7531, pp.146-155 (2012)
7. Cederquist, J.G., and Torabi Dashti, M.: An intruder model for verifying liveness in security protocols. In: Proceedings of FMSE '06, pp.23-32 (2006)
8. Desnitsky, V., Kotenko, I., and Chechulin, A.: Configuration-based approach to embedded device security. In: LNCS, Vol. 7531, pp.270-285 (2012)

9. Dick, N., and McCallum, N.: High-speed security Embedded security. In: Communications Engineer journal, Vol. 2, No. 2, pp.37-39 (2004)
10. Eisenring, M., Thiele, L., and Zitzler, E.: Conflicting criteria in embedded system design. In: IEEE Design & Test of Computers journal, Vol.17, No. 2., pp.51-59 (2000)
11. Feigenbaum, J., Freedman, M.J., Tomas, S., and Shostack, A.: Privacy Engineering for Digital Rights Management Systems. In: Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management, pp.76–105 (2001)
12. Gogniat, G., Wolf, T., and Burleson, W.: Reconfigurable Security Primitive for Embedded Systems. In: Proceedings of International Symposium on In System-on-Chip, pp. 23-28 (2005)
13. Grand, J.: Practical Secure Hardware Design for Embedded Systems. In: Proceedings of the 2004 Embedded Systems Conference, San Francisco, California, April 1 (2004)
14. Hedin, D., Sabelfeld, A.: A Perspective on Information-Flow. In: summer school Control Tools for Analysis and Verification of Software Safety and Security, Marktoberdorf, Germany (2011)
15. Juengst, W.E., and Heinrich, M.: Using Resource Balancing to Configure Modular Systems. In: Intelligent Systems and their Applications, IEEE Computer Society, Vol. 13, Issue 4, pp.50-58 (1998)
16. Knezevic, M., Rozic, V., and Verbauwhede, I.: Design Methods for Embedded Security. In: Telfor Journal, Vol. 1, No. 2 (2009)
17. Kocher, P., Lee, R., Mcgraw, G., and Ravi, S.: Security as a new dimension in embedded system design. In: Proceedings of the 41st Design Automation Conference (DAC ’04), pp.753-760 (2004)
18. Kommerling, O., and Kuhn, M.G.: Design principles for tamper-resistant smartcard processors. In: Proceedings of the USENIX Workshop on Smartcard Technology, pp.9–20 (1999)
19. Koopman, P.: Embedded System Security. In: IEEE Computer, No. 7 (2004)
20. Kotenko, I., and Polubelova, O.: Verification of Security Policy Filtering Rules by Model Checking. In: Proceedings of IEEE Fourth International Workshop on "Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications" (IDAACS'2011), pp.706-710 (2011)
21. Object Management Group, The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems, Version 1.1 (2011)
22. Moyers, B.R., Dunning, J.P., Marchany, R.C., and Tron J.G.: Effects of Wi-Fi and Bluetooth Battery Exhaustion Attacks on Mobile Devices. In: Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS'10), IEEE Computer Society, pp.1-9 (2010)
23. Pieters, W., and Coles-Kemp L.: Reducing normative conflicts in information security. In: Proceedings of the 2011 workshop on New security paradigms workshop, pp.11-24 (2011)
24. Pistoia, M., Chandra, S., Fink, S., and Yahav, E.: A Survey Of Static Analysis Methods for Identifying Security Vulnerabilities In Software Systems. In: IBM Systems Journal (2007)
25. Rae, A. J., and Wildman, L.P.: A Taxonomy of Attacks on Secure Devices. In: Australian Information Warfare and IT Security, 20–21 November 2003, Australia, pp.251–264 (2003)
26. Rae, A., and Fidge, C.: Identifying Critical Components during Information Security Evaluations. In: Journal of Research and Practice in Information Technology, pp. 391–402 (2005)
27. Ravi, S., Raghunathan, A., Kocher P., and Hattangady S.: Security in Embedded Systems: Design Challenges. In: ACM Transactions on Embedded Computing Systems,Vol.3,No.3, pp.461-491 (2004)
28. Ruiz, J. F., Harjani, R., Maña, A., Desnitsky, V., Kotenko, I., and Chechulin, A. A Methodology for the Analysis and Modeling of Security Threats and Attacks for Systems of Embedded Components. In: Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP2012). Munich, Germany, February 15-17 (2012)
29. Ruiz, J.F., Rein, A., Arjona, M., Mana, A., Monsifrot, A., and Morvan, M.: Security Engineering and Modelling of Set-Top Boxes. In: Proceedings of BioMedical Computing (BioMedCom), pp.113-122, 2012 ASE/IEEE International Conference (2012)
30. Sabin, D., and Weigel, R.: Product configuration frameworks-a survey. In: Intelligent Systems and their Applications IEEE Computer Society, Vol.13, Issue 4, pp.42–49 (1998)
31. SecFutur. Design of Secure and energy-efficient embedded systems for Future internet applications, FP7 Project Web site, http://www.secfutur.eu
32. Sprintson, A., El Rouayheb, S., and Georghiades, C.: A New Construction Method for Networks from Matroids. In: Proceedings of the 2009 Symposium on Information Theory (ISIT'09) (2009)
33. Wang, Z., Johnson, R., Murmuria, R., and Stavrou, A.: Exposing Security Risks for Commercial Mobile Devices. In: Computer Network Security, LNCS, Vol.7531, pp.3–2 (2012)
34. Wei, G., and Qin, Y.: An Approach of Product Configuration Based on Decision Tree and Minimum Conflicts Repair Algorithm. In: Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering (ICII '09),Vol.1, pp.126-129 (2009)
35. Yu, B., and Skovgaard, H.J.: A Configuration Tool to Increase Product Competitiveness. In: IEEE Intelligent Systems 13, Vol. 4, pp.34-41 (1998)