J Electr Eng Technol.2017; 12(5): 2051-2066
http://doi.org/10.5370/JEET.2017.12.5.2051

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

Filtering and Intrusion Detection Approach for Secured Reconfigurable Mobile Systems

Rim Idriss†, Adlen Loukil*, Mohamed Khalgui*, Zhiwu Li* and Abdulrahman Al-Ahmari****

Abstract – This paper deals with reconfigurable secured mobile systems where the reconfigurability has the potential of providing a required adaptability to change the system requirements. The reconfiguration scenario is presented as a run-time automatic operation which allows security mechanisms and the addition-removal-update of software tasks. In particular, there is a definite requirement for filtering and intrusion detection mechanisms that will use fewer resources and also that will improve the security on the secured mobile devices. Filtering methods are used to control incoming traffic and messages, whereas, detection methods are used to detect malware events. Nevertheless, when different reconfiguration scenarios are applied at run-time, new security threats will be emerged against those systems which need to support multiple security objectives: Confidentiality, integrity and availability. We propose in this paper a new approach that efficiently detects threats after reconfigurable scenarios and which is based on filtering and intrusion detection methods. The paper's contribution is applied to Android where the evaluation results demonstrate the effectiveness of the proposed middleware in order to detect the malicious events on reconfigurable secured mobile systems and the feasibility of running and executing such a system with the proposed solutions.

Keywords: Mobile system, Android, Reconfiguration and adaptation, Security, Filtering and detection, Multi-agent system, Modelling and evaluation

1. Introduction

With the increasing of network connectivity and computing power, we have seen significant propagation in using mobile systems [50] which become more open. Today, more and more services and platforms have been developed and deployed on these systems, such as [20] software as a service (SaaS), mobile payment and ticking, pervasive information and content sharing (audio, text, and video), voice-over-IP (VoIP) applications and many more. These evolutions come with new security challenges that are different and cannot be satisfied with the security requirements of conventional limited purpose systems.

Although there are several security solutions such as filtering and detection, we noticed an increasing number of vulnerabilities and cyber-attacks, especially the system behavior modification at run-time which can be a manner to run different types of attacks. We look to resolve this problem in [3] in which we propose a new multi-agent architecture software solution to allow feasible secured reconfigurations of mobile devices. Nevertheless, the mobile systems are characterized by real-time constraints where temporal requirements are critical and the choice of security mechanisms is important to satisfy real-time constraints. In fact, added security mechanism may cause unacceptable security level degradation or deadline missing. To address this problem, we proposed in [4] a middleware based on an execution model of tasks and security solutions while guaranteeing an optimum level of security and meeting the deadlines.

Nowadays, several techniques have been proposed and used in [7, 22] for improving the security level and to detect the possible intrusions of any unusual behaviors. These techniques include filtering and intrusion detection systems (IDS). With the development of reconfigurable systems, there are new attack vectors which can be used to compromise these techniques while benefiting of these limits. Unfortunately as presented in [44], these solutions are inappropriate for dynamic reconfigurations in mobile devices regarding their limited computing and communication capabilities. For this reason, we propose to implement a new intelligent layer based on filtering and detection in order to improve the control of the tasks and the dynamic events. The proposed layer is composed of four components: (i) Collect, (ii) Analysis, (iii) Detection, and (iv) Filtering. This layer is added to the middleware that we propose in [3, 4]. This middleware is compsosed of four agents: (i) Reconfiguration agent which handles automatic

† Corresponding Author: School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), China./INSAT Institute, University of Carthage, Tunisia
* INSAT Institute, University of Carthage, Tunisia
** Industrial Engineering Department, College of Engineering, King Saud University, Riyadh 12372, Saudi Arabia.
*** Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau / School of Electro-Mechanical Engineering, Xidian University, China
Received: August 15, 2016; Accepted: May 10, 2017

reconfiguration scenarios, (ii) Security agent which guarantees safe reconfigurations, (iii) Scheduling agent which verifies real-time constraints, and (iv) Execution agent which is responsible for the management of the system memory. Our contribution is based on the relation between the security agent and the detection as well as filtering layer in order to enhance the security quality using the following security mechanisms: (i) Firewall and (ii) IDS. The contribution made in this research is applied to Android to evaluate the performance of the proposed solutions. We aim in our experimentation to check the degree of security after each reconfiguration scenario that should also guarantee the real-time feasibility of the new and old tasks as well as mechanisms.

This paper is structured as follows. Section II demonstrates the necessary background needed in the context of this work and proposes the current approach through a discussion of the related works. Section III describes the formalization of the secured reconfigurable systems. In Section IV, we explain the proposed architecture. Then, we present the case study, the proposed algorithms, the implementation and also the evaluation of the proposed solution in Section V. Finally, Section VI concludes this paper and discusses the future work.

2. Background and State of the Art

This section brings up a description of the context of the work. It deals with the known security attacks against the reconfigurable mobile systems. We also describe the various countermeasures which can be taken against these attacks.

2.1 Reconfigurable mobile systems

Mobile Systems described in [25, 14] are becoming more popular because of the increase in their mobility aspect, processing power and personal nature. In order to understand and define the possible reconfigurations, the current state of mobile devices presented in Fig. 1 [16] needs to be studied. This architecture is composed of different interacting layers based on a middleware. It presents the application framework layer which provides an API to permit the interactions between hardware components applications. The virtualization is designed specifically to run on limited hardware where a swap space does not exist. The lowest layer performs low-level functionalities (power, process and memory management, etc.) and contains hardware drivers.

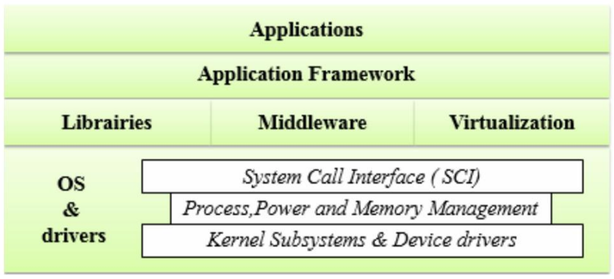

Applications
Application Framework
Librairies
Middleware
Virtualization
OS & drivers
System Call Interface (SCI)
Process,Power and Memory Management
Kernel Subsystems & Device drivers

Fig. 1. Software-based architecture

Mobile embedded systems offer several mechanisms to permit the control of devices by reading and interpreting data from sensors [19-51]. The designers can optimize those systems to increase performance with a required reconfigurablity, and reduce the size of the product. For these reasons, several interesting researches have been dedicated to develop those reconfigurable architectures [1]. To allow the addition-removal-update of services, reconfiguration scenarios [47] should be on-line or off-line operations, and this in order to adjust the system to its environment under user requirements. For example, it is possible to add an external memory to the device.

Consequently, the corresponding architecture and tasks will be changed. We can identify several ways to build a reconfigurable system [45, 46]: (i) Manual reconfiguration which requires the user intervention. In this case, when the user starts a new service, the system will be reconfigured according to the selected service. This type of reconfiguration is related to the higher layer (Application), (ii) Automatic reconfiguration which is based on well-defined conditions where the starting of a service requires the activation of another one. In this case, the system must be reconfigured automatically without the user intervention. In new mobile embedded technologies, the characteristics of reconfigurable systems lead to new challenges [15, 43] that need to be considered (for example processing performance, flexibility, network connectivity and power consumption, etc.). As reconfigurable systems become more popular, the concerns arise about their security and privacy [16-49-52]. Although, the automatic reconfigurations provide a required flexibility of hardware and software components, they may compromise security and integrity of the embedded system design. Therefore, the contribution of this paper is to guarantee a secured system after any run-time reconfiguration.

2.2 Security mechanisms

Mobile embedded systems are vulnerable to many operational problems and intentional attacks due to their characteristics [11]: (i) Limited processor utilization, (ii) Memory capacity, and (iii) Cost sensitivity. Nowadays, several studies are reported to deal with the security of these systems such as the work in [21]. These works present the vulnerabilities of each component (software) and some security mechanisms that can provide critical functions which could be inhabited by external events. Nevertheless, they might not satisfy reconfigurable real-time requirements. Each system should provide classically basic security functions [9] such as: (i) Confidentiality that guarantees the secrecy of data transmission, (ii) Integrity is against the unauthorized modification of data (including accidental change, destruction or alteration), (iii) Availibity is the process of assuring that information services will be ready for use when expected, and (iv) Authentication which determines the identity of a communication part. The goal of security services is to protect the systems from misbehavior threats [32] which can be classified by the means to launch the attack [8]. We present the logical attacks for reconfigurable systems [24]: (i) Interception-based attacks: In this type of attacks, the hacker tries to eavesdrop sensitive data in order to compromise the confidentiality of exchanged data. For example, it could use a logical analyzer for inner line bus probing of a mobile device, (ii) Interruption-based attacks: Those attacks are aimed the availability of mobile systems by making them unusable. This could be achieved through an attack named energy exhaustion which exploits the energy resource in battery-powered systems. Besides, the attacks named denials of service are very common in reconfigurable mobile devices with limited processor utilization and storage capacity where the mobile device can be failed in the event of intensive workload, and (iii) Modification-based attacks: Those attacks compromise the integrity of mobile systems by exploiting vulnerabilities. Besides, those vulnerabilities can allow an attacker to write memory at arbitrary position. Thus, it is possible to modify a return address or a function pointer [21].

In this paper, we are interested in the basic security solutions based on the detection and filtering methods. Beginning with the filtering mechanisms, Fig. 2 shows the firewall architecture which checks the traffic going from or coming to the system. As a result, it protects all communications from outside to the local network and blocks anything that could be harmful [26]. The packet-filter firewall works between the network layer and the transport layer [30].

It ensures a network security by filtering TCP/IP communications. The firewall examines the TCP/IP headers in order to decide if the packets are allowed or denied. Packet-filtering makes decisions based upon the following header information: (i) The source and destination IP address, (ii) The network protocol, and (iii) The TCP or UDP source and destination port.

The second important technique is the intrusion detection presented in Fig. 3.

The intrusion detection system continuously monitors and collects mobile system data in order to extract various information obtained from the system such as the battery consumption, network traffic, and CPU usage. There are several different types for IDSs: (i) Host-based IDS (HIDS), and (ii) Network-based IDS (NIDS). Some examples of these IDSs types are described in [7-10-12]. The main problem of the intrusion detection systems is the CPU, memory and energy consumption since they are running with limited ressources. Those systems are extremely challenging to design and it should be done on a low-level OS as much as possible under memory constraints.

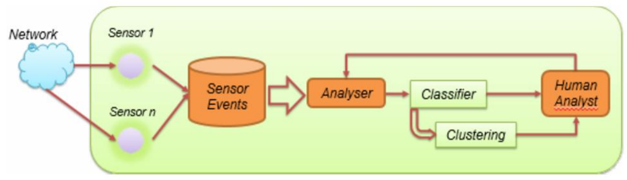


Fig. 2. The firewall architecture

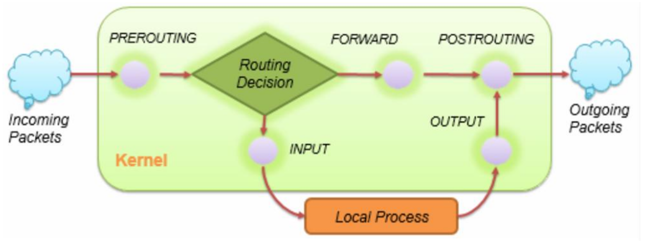


Fig. 3. The IDS architecture

2.3 State of the art: secured reconfigurable mobile systems

Nowadays, several research approaches deal with secured mobile systems, some of them deal in particular with secured reconfigurable mobile systems. The work in [13, 48] proposes a new approach in order to detect malware targeting mobile devices. This approach monitors continuously a time-stamped security data within the target mobile device and processes by the KBTA methodology (knowledge-based temporal abstraction). Besides, the security in dynamic reconfiguration is studied in [18]. This article describes an approach for SLA aware dynamic reconfiguration which has been captured as IRAILS pattern. A case study is also presented to demonstrate the various ways of dynamic reconfigurations. The proposed method in [22] is based on a software architecture for a server platform that allows secured bidirectional communication using TCP/IP with reconfigurable logic devices. In [23], a method is proposed to secure a partially reconfiguration based on the encryption and authentication mechanisms. Nowadays, there has been a considerable amount of research works that deal with the filtering and detection intrusion in mobile systems [10]. These researches include different approaches: (i) dynamic and static approaches [29], (ii) machine learning and sandboxing techniques [40], and (iii) data-mining based methods [32]. The work reported in [39] proposes a run-time analysis system for Android malware detection. This system consists of three components: event creator, parser, and log monitor. The event creator allows the simulation of the user's action through the result of the static analysis. The proposed framework reported in [38] is the AppsPlayground for Android that automates the analysis smartphone applications. This framework integrates multiple components that comprise different automatic exploration and detection techniques. A multi-agent architecture is proposed in [34] to create Android profiles for the monitoring of SMS services by applying multiple detection approaches. In [35], an agent-based framework is proposed to protect Android devices where the different agents can be dynamically launched. The analytical components in this framework implement machine-learning algorithms. The work reported in [36] presents a real-time host and network mobile agent based intrusion detection system (HNMAIDS). This system detects any suspicious behavior on the network/software by using an agent that sends an alert to the network administrators who can prevent intrusions. Besides of filtering method, the work reported in [31] proposes a firewall-based solution for protecting Android against different attacks. In [37], a filtering system is proposed to achieve a similarity comparison scheme based on the App feature information such as permissions, image resources and directory structures of the APK file (an Android App file named "ApplicationName.apk").

Although these studies are exciting, we aim to improve the filtering and detection method for the security of reconfigurable mobile systems. We propose a new layer for the execution of the firewall and IDS when a new reconfiguration scenario is applied. In this case, some corresponding properties (deadlines, security level, etc.) can be violated. To allow secured reconfiguration scenarios while meeting security and temporal constraints, we propose an intelligent layer named filtering and detection layer. The proposed approach reduces the security risk of tasks and the execution of mechanisms after any scenario by overcoming the limitations of the firewall and IDS.

3. Formalization of Secured Reconfigurable Mobile Systems

We consider a mobile system denoted by *Sys* that will be reconfigured automatically. Each reconfiguration scenario is assumed to be a run-time addition-removal-update of OS tasks [41, 42]. The system Sys={ τ_1 ; τ_2 ... τ_n } is characterized by a subset of n OS tasks that is executed at a particular time *t*. Each periodic task denoted by τ_i(where $i \in [1..\, n_p]$) is essentially characterized by its: (i) Deadline D_i is the time that τ_i must be accomplished, (ii) Period T_i which represents the interval between consecutive events τ_i. In the case of an aperiodic event, the concept of period is totally not listed, (iii) Computation time C_i is the worst case execution time (WCET), and finally (iv) Security level SL_{τ_i} presents the degree of violating security constraints. The CPU utilization U_{per} of n_p periodic tasks is given by:

$$\boldsymbol{U_{per} = \sum_{i=1}^{n_p} \frac{C_i}{T_i}} \quad (1)$$

An aperiodic task is defined also by τ_i (C_i, D_i, SL_{τ_i}) where $i \in [1..\, n_p]$. The aperiodic processor utilization U_{aper} is determined by the average service rate λ_c and the average arrival rate λ_r based on the *M/M/1* model [5-2]. The aperiodic events arrive according to the Poisson process with a Poisson distribution. Consequently, the processor utilization U_{aper} for n_{ap} tasks is formalized by:

$$\boldsymbol{U_{aper} = \lambda_r / \lambda_c} \quad (2)$$

To handle secured reconfiguration scenarios, we assume a set of security mechanisms denoted by Set_M={M_1,..,M_{n_m}}. The *j*–th mechanism is considered as a real-time task $M_j(T_j, C_j, SL_j)$. The CPU utilization U_M of n_m mechanisms is denoted by:

$$\boldsymbol{U_M = \sum_{j=1}^{n_m} \frac{C_j}{T_j}} \quad (3)$$

The total processor utilization U_{Sys} of *n* tasks:

$$\boldsymbol{U_{Sys} = U_{per} + U_{aper} + U_M} \quad (4)$$

The total processor utilization before the *k*–th reconfiguration scenario at run-time by adding/removing some periodic/aperiodic tasks or mechanisms at t is denoted by $U_{sys}^{bef}(t)$ and after the reconfiguration is denoted by $U_{sys}^{aft}(t)$. Each mechanism $M_j$is characterized by different services: availability, confidentiality and integrity [6]. In fact, a security service can be executed by one or more mechanisms. According to [9], we assume a subset of security services denoted by Set_{Ser} = {A, I, C}: (i) Availability service, (ii) Integrity service, and (iii) Confidentiality service. We set for each security mechanism a security level which presents a combination of message and transport. This parameter may be high]2,3], medium [1,2], or low [0,1]. Thus, the security control of each mechanism M_j ($j \in [1...\, n_m]$) is formalized by:

$$\boldsymbol{SL_{M_j} = \begin{pmatrix} SL_{jC} \\ SL_{jI} \\ SL_{jA} \end{pmatrix}} \quad (5)$$

We defined a new metric named the risk denoted by *Rk* in the paper [3] depending on the security level for securing the reconfigurability as follows:

$$\boldsymbol{RK = SL/SC} \quad (6)$$

Where SL presents the security level of violating security constraints of tasks and SC presents the security controls (3) defined by:

$$\boldsymbol{SC = \sum_{j=1}^{n_m} SL_{M_j}} \quad (7)$$

We assume also a threshold Se_{HR} of the risk that does not allow the system to exceed the security constraints. Besides, we define for a low security level a threshold denoted by Se_{LR}. Thus, we determine the risk of any reconfiguration scenarioRk_r^j, at any time *t* before applying the reconfiguration, by this expression:

$$\boldsymbol{Rk_r^j} = \sum_{i=1}^{n'} \boldsymbol{RK(\tau_i)} + \sum_{i=1}^{n^{add}} \boldsymbol{RK(\tau_i)} + \sum_{i=1}^{n^{del}} \boldsymbol{RK(\tau_i)} \quad (8)$$

Where (i) n' is the number of OS tasks and mechanisms at *t*, (ii) n^{add} presents the added tasks to *Sys* at *t*, and (iii) n^{del} presents the tasks that will be removed at *t*.

4. Filtering and Intrusion Detection Approach for Secured Reconfigurable Mobile Systems

This section presents the architecture of the proposed approach that provides new concepts and methods to deal with security issues of reconfigurable mobile systems.

4.1. Proposed approach

We propose to implement an intelligent layer based on the filtering and intrusion detection in order to improve the control of tasks and dynamic events. The proposed layer is composed of four components: (i) Collection, (ii) Analysis, (iii) Detection, and (iv) Filtering. This approach is explained in Fig. 4. This layer is added to the middleware that we propose in [3,4]. This middleware is composed of (i) Reconfiguration agent which handles automatic reconfiguration scenarios, (ii) Security agent which guarantees safe reconfigurations, (iii) Scheduling agent which verifies real-time constraints, and (iv) Execution agent which is responsible for the management of the system memory. This middleware is based on the execution model of real-time tasks and security mechanisms while meeting the related deadlines and minimum levels of security. This model is essentially based on two parameters named (α, β). It defines respectively the execution period and the occurrence of security solution M_j. A security mechanism can be implemented many times that may lead to deadline missing which can produce failures. As a result, we propose to determine β the occurrence number of these mechanisms in order to stabilize the system. However, it can be executed only once for various tasks by determining the α parameter. The scheduling agent receives the new subset of tasks including the subset of mechanisms that is considered as a simple task: *Sys* = $\{\tau_1, \tau_2..\tau_n, M_1,, M_{n_m}\}$, where n presents the number of aperiodic/periodic tasks and n_m presents the number of security mechanisms.

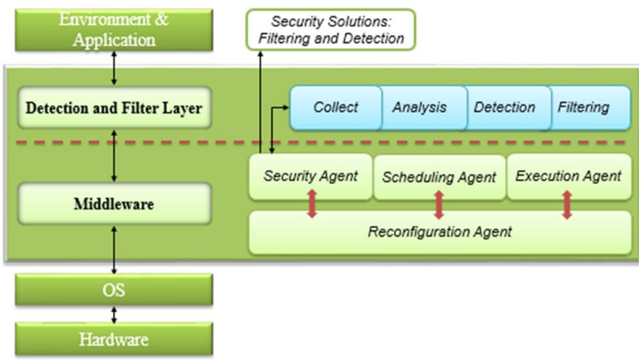


Fig. 4. Architecture of the proposed approach

When a reconfiguration scenario is applied automatically at run-time to add new security mechanisms or new real-time tasks, we can conclude that $U_{sys}^{aft}(t)$ is equal to or greater than $U_{sys}^{bef}(t)$, with:

$$U_{Sys} = \sum_{i=1}^{n} \frac{C_i}{T_i} + \sum_{j=1}^{n_m} \frac{\beta_j C_j}{\alpha_j T_j} \quad (9)$$

The formula (8) calculates the CPU U_{Sys} of the mobile system that is the set of *n* real-time tasks and n_m mechanisms. The periodicity of mechanisms is adjusted by a scaling factor in order to define a new period αT_j:

$$\frac{U_M}{1-U_T} \leq \alpha_j \leq \frac{1}{U_M} \sum_{j=1}^{n_m} C_j \quad (10)$$

Besides, each mechanism can be implemented many times, which may lead to failures. So, we propose to determine the number of occurrences denoted by β_j of each mechanism:

$$\boldsymbol{\beta_j = Occurrence(M_j)\ where\ 1 \leq \beta_j \leq n_m} \quad (11)$$

The Detection and Filter Layer, shown in Fig. 4, is based on the methodology of risk management [6]. It communicates only with the security agent. Beginning with the first component "Collection of information" defined by *CL(Sys(t))* :

$$CL(Sys(t)) = \begin{Bmatrix} \xi_l(t) \\ \xi_{nw}(t) \end{Bmatrix} \quad (12)$$

The detected set of tasks at *t* (*Sys(t)*) routed through the security agent to this layer is classified into two groups: (i) those that do not require a network connection at *t* $\xi_l(t)$ and (ii) those which are related to the network at *t* $\xi_{nw}(t)$. The tasks belonging to the first group are characterized by the anonymity which means that it is a new task and has a high risk for the system. To assess the risk of those tasks, the system will run them in a sandbox [17]. During the execution, we can identify the required permissions to execute each task. These permissions are divided into three types: (i) Dangerous, (ii) Normal, and (iii) Undangerous. After the identification of the type of permission, the security agent decides the adequate solution (block or accept the execution of this task). Nevertheless, the task of the second group presents a high risk since the internet connection brings more threats. The second component is the analysis of the collected information denoted by *An(t)*. Each packet will be analyzed in order to determine the useful information:

$$\boldsymbol{An(t) = Data(CL(Sys(t)))} \tag{13}$$

Where *Data* presents the source and destination ports, source and destination IP addresses, data and protocol.

Based on the collected *CL(Sys(t))* and analyzed information *An(t)*, we can detect the probable intrusions by using data-mining denoted by *DI(t)*. This technique presents the set of statistical and mathematical methods to analyze and extract the knowledge from a data set. Several methods are used in this approach and we have chosen to use the concept of the decision trees because we can classify the data into two homogenous groups ("normal" and "abnormal").

$$DI(t) = DMining(An(t)) = \begin{Bmatrix} normal \\ abnormal \end{Bmatrix} \tag{14}$$

After the generation of rules, the arrived packet to the "Intrusion Detection" module is checked and verified by these rules in order to determine in particular whether it is an intrusion or not. This classification is routed to the component of "Filtering" in order to determine the corresponding decision.

To state the problem mathematically, we define a basic model of each reconfiguration scenario denoted by $\mathrm{Rg_k}$ described in (15). This model is based on eight rules that must be applied and tested. For each reconfiguration scenario *k*, we formulate the system *Sys* with a set of periodic and aperiodic tasks. The plugin must calculate the processor utilization $U_{sys}^{bef}(t)$ of the detected tasks at *t* ($\mathrm{R_1}$) and also the risk of applying this scenario *Rk* ($\mathrm{R_2}$). Based on the results of those rules, the detection and filter layer run through the three levels ($\mathrm{R_3}$) and extract the *CL*(*Sys*(*t*)), *An*(*t*) and *DI*(*t*). So, a set of mechanisms Set_M will be proposed and added to the Sys' at *t* ($\mathrm{R_4}$).

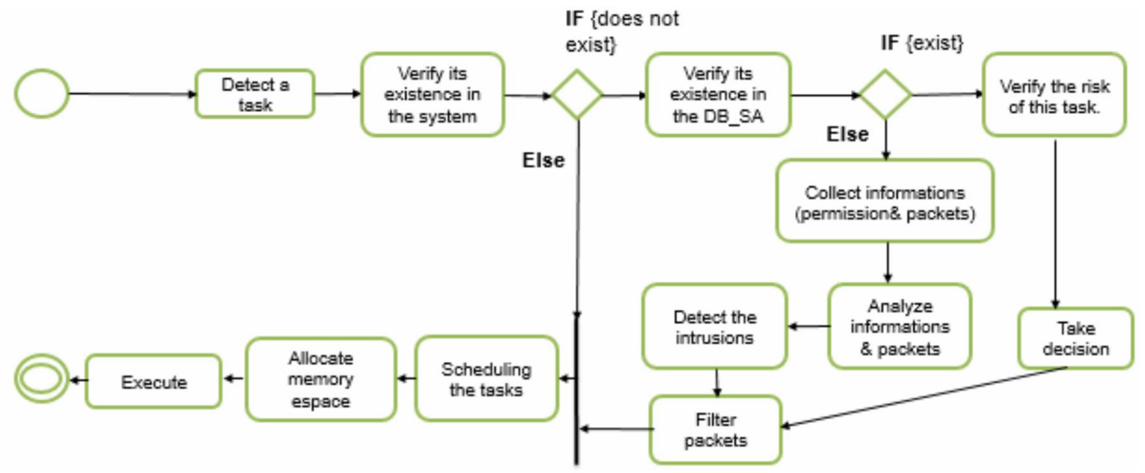
$$Rg_k \begin{Bmatrix} (R_1)\colon U_{sys}^{bef}(t) = U_{per} + U_{aper} + U_M \\ (R_2)\colon Rk = SL/SC \\ (R_3)\colon CL(Sys(t)), An(t), DI(t) \\ (R_4)\colon Set_M = \{M_1, M_2, \dots, M_{n'_m}\} \\ (R_5) : \beta_j = Occurrence(M_j) \\ (R_6)\colon \frac{U_M}{1-U_T} \leq \alpha_j \leq \frac{1}{U_M} \sum_{j=1}^{n'_m} C_j \\ (R_7)\colon Se_{LR} \leq Rk_r^j \leq Se_{HR} \\ (R_8)\colon U_{sys}^{aft}(t) \leq 1 \end{Bmatrix} \tag{15}$$

We aim in this paper to determine a minimum processor utilization ($\mathrm{R_5}$) and ($\mathrm{R_6}$) such that each task satisfies its constraints by using the parameters β and α. Finally, the proposed solution must guarantee a feasible system with $\mathrm{Se_{LR}} \leq Rk_j^r \leq \mathrm{Se_{HR}}$ ($\mathrm{R_7}$) and $U_{sys}^{aft}(t) \leq 1$ ($\mathrm{R_8}$).

4.2 Communication protocol

We define in this section the communication protocol (shown in Fig. 5) between the two layers (the middleware, the detection and filter layer), where SA manages the various requests and takes the appropriate decision (denied requests, authorized requests, or other).

Addressing the needs of the user requirements, the reconfiguration agent checks the evolution of the environment and applies the adequate reconfiguration scenarios at the different levels (structure, architecture or data). For each scenario, RA should send a request to SA and determine the criteria to be secured. The security level of the system is also determined in the case of this reconfiguration form. In this case, if the added tasks present a high risk, then it forwards the message to the



IF {does not exist}
IF {exist}
Detect a task
Verify its existence in the system
Verify its existence in the DB_SA
Verify the risk of this task.
Else
Else
Collect informations (permission& packets)
Detect the intrusions
Analyze informations & packets
Take decision
Execute
Allocate memory espace
Scheduling the tasks
Filter packets

Fig. 2. Communication protocol of the overall system

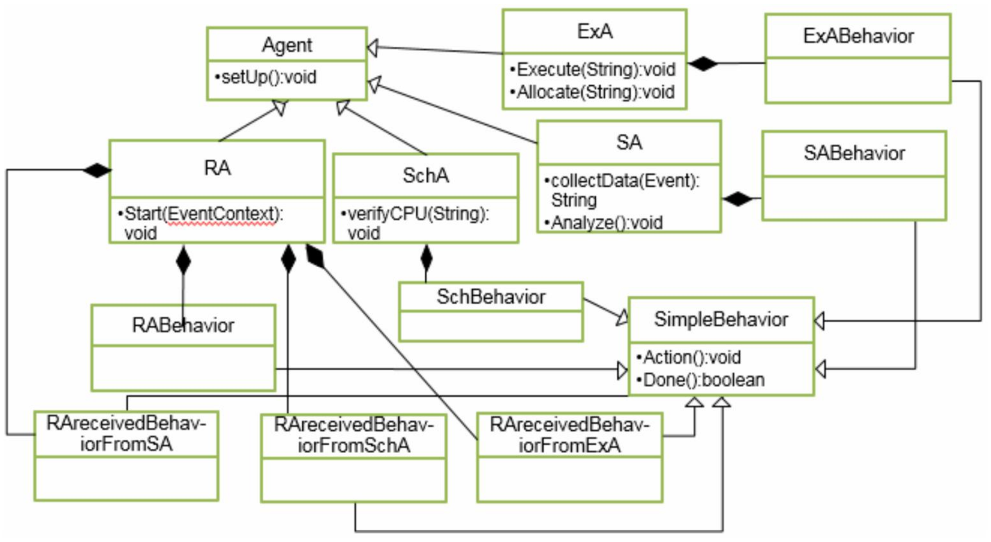


Fig. 3. Agents class diagram

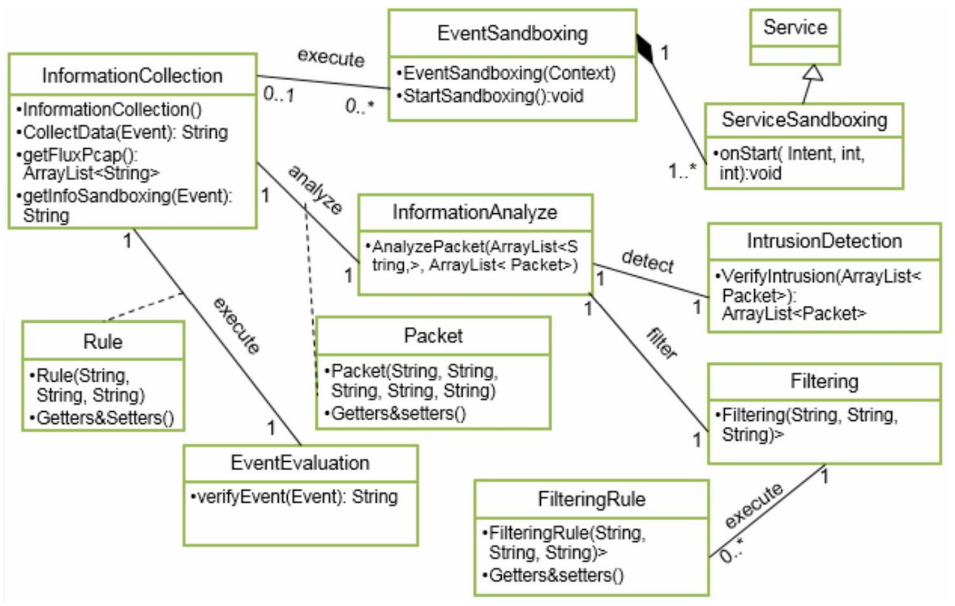

InformationCollection
•InformationCollection()
•CollectData(Event): String
•getFluxPcap(): ArrayList<String>
•getInfoSandboxing(Event): String
execute
0..1
0..*
EventSandboxing
•EventSandboxing(Context)
•StartSandboxing():void
1
Service
ServiceSandboxing
•onStart(Intent, int, int):void
1..*
analyze
1
InformationAnalyze
•AnalyzePacket(ArrayList<String,>, ArrayList< Packet>)
detect
IntrusionDetection
•VerifyIntrusion(ArrayList< Packet>): ArrayList<Packet>
Rule
•Rule(String, String, String)
•Getters&Setters()
execute
Packet
•Packet(String, String, String, String, String)
•Getters&setters()
filter
Filtering
•Filtering(String, String, String)>
EventEvaluation
•verifyEvent(Event): String
FilteringRule
•FilteringRule(String, String, String)>
•Getters&setters()
execute
0..*

Fig. 4. Detection and filter class diagram

filtering and detection layer to verify it. However, the response of the three agents is controlled by the SchA in order to meet the real-time constraints.

4.3 UML design

We present in this section the class diagram of the proposed solution. Starting with the reconfiguration layer diagram presented in Fig. 6, this layer is based on the concept of agents presented by the principle package "Agents". Every proposed agent (*RA*, *SA*, *SchA* or *ExA*) inherits from this package under the platform JADE. For each class (Agent), a behavior named "SimpleBehaviour" is defined.

We present also in Fig. 6 the class diagram of the plug-in responsible for the detection of a new task or a new event. We show in Fig. 7 the package of the filter and detectionlayer. This package ensures the requirements of the filtering layer according to the need of SA. The Sandboxing solution is defined by the class "Service-Sandbox" through the class "EventSandboxing". The event processing is treated according to the rules ("Rule" determined by the class "EventEvaluation").

5. Experimentation

In this section, we present a case study in order to expose the paper's problem, explain the proposed algorithms and expose the implementation.

5.1 Case study

By considering an Android-based smartphone [33, 27], this work will study the security aspects of its real-time automatic reconfigurations. A reconfiguration scenario is assumed to be run-time additions-updates-removals of OS tasks. Considering CPU, memory and data storage constraints in Android [28, 29], the system Sys is characterized by a set of OS tasks at a particular time t.

In this research paper, we propose to classify three behavioral modes of the devices: (i) Phone calls and camera, (ii) Surfing on internet, financial transactions and (iii) Download/update of applications. We define the following groups of tasks to offer all the required services

(*Numberservices* = 3) where we find some periodic and aperiodic tasks:

• **Group 1:** (Device functions: phone-calls, camera, etc.): τ_1 is based on an input event to activate a key on the smartphone. τ_2 blocks numbers from a particular geographic area (using the country codes) where the call can be redirected directly to the mail or directly stopped,

• **Group 2:** (Surfing outside): τ_3 is the OS task that can send a file via bluetooth, τ_4 is a financial task allowing a secured communication with banks for a payement service, τ_5 is a OS task that allows a secured remote payement of bills and τ_6 is an access task to a remote website.

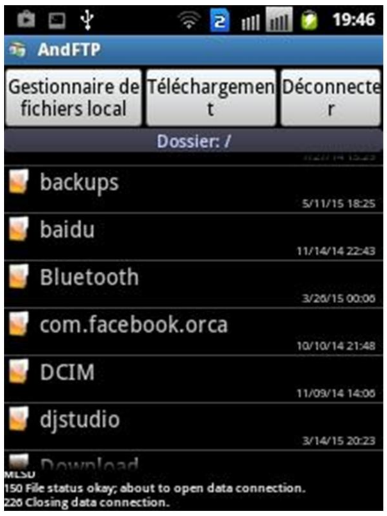
• **Group 3:** (Download and update of smartPhone applications): τ_7 is an update secured task and τ_8 presents another topic of tasks that present all activity related to the execution of an external codes or the setting of the smartphone such as processor, memory, battery, etc. We propose two other tasks τ_9 and τ_{10} which present respectively the access to data through the FTP and Telnet flow. To handle reconfiguration scenarios, we suppose a set
of mechanisms denoted by $Set_M=\{M_1 \ \ M_2, M_3\}$ that present respectivily: (i) Authentication, (ii) Firewall, and (iii) Intrusion detection system. We assume also ten tasks $\{\tau_1, \tau_2, ..., \tau_{10}\}$. For example, if the system requires the addition of the task τ_6 to have an access to a remote website thanks to an integrity mechanism which is not uploaded, then it should be added with τ_6. After that, if we want also to add the task τ_4 hanks to integrity and confidentiality mechanisms such that the second is not uploaded, then the confidentilaity mechanism should be added. We present in Table 1 some reconfiguration scenarios to be applied at runtime in order to adapt the system according to its environment.

We assume also in this table the security levels of the different tasks and mechanisms to be applied. To determine the risk of each scenario Rk_j^r, we use the different equations (3) and (4) defined in section 3. The first scenario Rg_1 uploads the tasks τ_1, τ_2 and τ_3 in order to offer the services (i) Control of key, (ii) Control of remote contacts, and (iii) Bluetooth communication with remote devices. The application of the last scenario adds the tasks τ_9 and τ_{10} with the security level 2.5. In this case, the mechanisms M_1 and M_2 are added. We suppose that the highest acceptable security level is 3 and the lowest is 1 (According to [6]). The system's security level is initially the sum of the levels of the different uploaded tasks (defined in section 4). The system Rg_1 at t_0 is feasible since the CPU utilization factor $U_{sys}^{bef}(t_0)$ is equal to 0.20 < 1. We assume a reconfiguration scenario Rg_3 by adding two tasks τ_4 and τ_5, and aperiodic task τ_6 with the distributions λ_c=0.5 and λ_r = 0.2 (Table 1). The following set of mechanisms (M_1, M_2 and M_3) will be added to the system *Sys0*. The CPU utilization factor $U_{sys}^{aft}(t)$ is equal to 1.383 > 1. The system after reconfiguration becomes infeasible where the tasks (τ_5 andτ_6) miss their deadlines. When the seventh reconfiguration scenario is applied, a hacker makes a sniffing on the network and retreives the login and the password sent by the user. Thus, this automatic event benefits the hacker to illustrate the smartphone victim's content (shown in Fig. 8), it can even download files, images, etc. These files can contain some passwords (bank cards, e-mail account, etc).

This example shows the paper's contribution that deals with secured reconfigurations based on existing mechanisms (Firewall and IDS). Indeed, some reconfiguration scenarios can bring critical mobile systems to unsafe states. Thus, the addition of the security mechanisms for each scenario does not guarantee a high security level because of the mechanisms's limits.

Table 1. Reconfigurations scenarios

Reconfiguration Scenarios: Rg_k	Added tasks (n_{add})	Security Level SL_{τ_i}	Computation Time C_i	Periods T_i	Deadline D_i	Security Mechanisms Set_M
$Rg_1 : \{\tau_1, \tau_2, \tau_3\}$	τ_1	0,1	5	60	10	-
	τ_2	0,2	2	55	15	-
	τ_3	0,7	3	45	10	-
$Rg_2 : \{\tau_1, \tau_2, \tau_3, \tau_4\}$	τ_4	2	2	40	8	M_1, M_2
$Rg_3 : \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$	τ_5	2,8	3	50	15	M_1,M_2,M_3,
	τ_6	1,1	8	-	15	
$Rg_4 : \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$	-	-	-	-	-	-
$Rg_5 : \{\tau_1, \tau_2, \tau_7, \tau_8\}$	τ_7	2	10	35	10	M_2,
	τ_8	1,7	20	-	20	
$Rg_6 : \{\tau_1, \tau_2, \tau_7, \tau_8, \tau_9\}$	τ_9	2	15	20	-	M_3
$Rg_7 : \{\tau_1, \tau_2, \tau_{10}\}$	τ_{10}	2.5	10	35	10	M_1, M_2, M_3



```
backups            5/11/15 18:25
baidu              11/14/14 22:43
Bluetooth          3/26/15 00:00
com.facebook.orca  10/10/14 21:48
DCIM               11/09/14 14:06
djstudio           3/14/15 20:23
Download
MLSD
150 File status okay; about to open data connection.
226 Closing data connection.
```

Fig. 8. Access attempt to FTP server

We propose an original layer added to the proposed middleware in order to improve the quality of the security assured by the Firewall and the IDS. This layer is proposed to evaluate the security level of the system after any reconfiguration scenario in order to decide if it is possible to apply it or not. This paper addresses a solution that does not only reconfigure the tasks and the mechanisms of the system, but also the evaluation of the existing solutions. The security level becomes in this paper a flexible parameter that evolves with the application of scenarios. The filtering and detection methods become able to detect the dynamic or automatic event without the intervention of the user.

5.2 Implementation

In this paper, we present the design of a multi-agent security platform on the Android version 4.1 with "Java Agent Development" (JADE) [5] through a middleware which complies with the FIPA specifications. A prototype system is succecfully implemented and studied. In our design, the agents in the prototype system adjust dynamically their behaviors to achieve a balance between security needs and the consumption. The reconfiguration agent is useful for real-time and secure reconfigurations of mobile systems. It checks the environment evolution before applying automatic reconfiguration scenarios. Nevertheless, when new tasks are added, the privacy properties of reconfiguration and also the behavior of the system may be violated. For this reason, this agent will control any new task and if there is a risk for applying an external object or entity, then it will apply the adequate solution presented by Algorithm 1: First, the mobile system reads the initial configuration containing the parameters of all the tasks *Sys*.

After that, *RA* checks the addition scenarios that add tasks to *Sys*(t) and then calculates the processus utilization $U_{sys}^{bef}(t)$ and the risk *Rk*. Those parameters are used to

```
Algorithm 1 : Implementation of the Reconfiguration Agent
Begin
  1.Input: Reconfiguration with $\xi(Sys)$;
  2.Modify all the tasks that have the highest security level with the highest set-priority;
  3.Calculate $Rk_{Bef}(Rg)$ ;
  4.If($Rk_{Bef}(Rg) \leq Se_{LR}$)
      Apply Reconfiguration with $\xi(Sys)$
   Else
   If(Load new architecture) or ($Rk_{Bef}(Rg) > Se_{LR}$)
      Request (RA, SA($\xi$ (Sys)), $Rg_i$);
      Output : $Rg_i$;
   EndIf End.
```

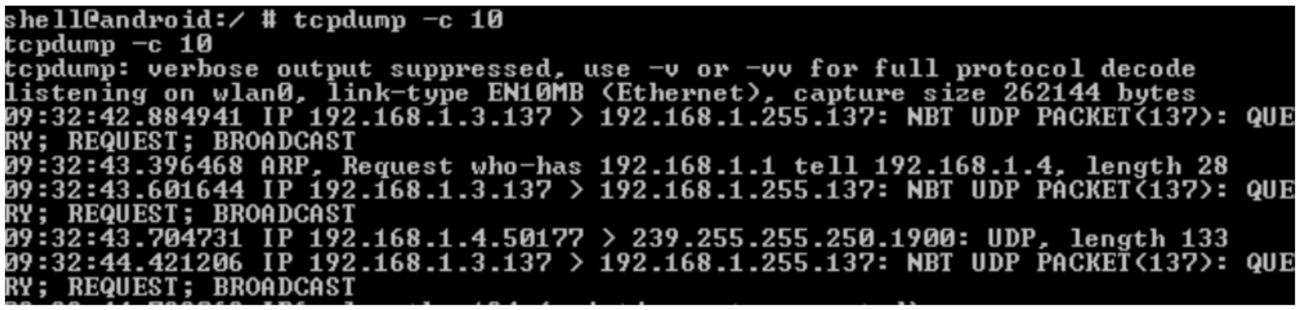
evaluate the solution and to limit the processus utilization. After each addition scenario, the agent reads the added tasks and assigns the old and new tasks with different priorities based on security level values. Thereafter, each added task must be verified if it belongs to the system or not. In the case of rejection, *RA* needs to receive a permission from the security agent *SA*.

We develop an Algorithm 2 to implement the proposed intelligent control security agent. This algorithm aims to possibly reduce the risk before and after any reconfiguration scenario. First, *SA* receives a set of tasks that present the scenario with their parameters and then verifies each task with the database *DBSA*. If the task exists in the Database, then the result should be analysed based on another database of security policy and mechanisms in order to add security solutions. Otherwise, *SA* will send a request to the filter and detection layer in order to examinate this event.

```
Algorithm 2 : Implementation of the Security Agent
Begin
Input: $Rg_k$
 For any Request($Rg_k$)
  1.$\xi(Sys) \leftarrow$ Interpreter($Rg_i$)
  2.If($\nexists\xi$(Sys) in $DB_{SA}$)
        Analyzer($\xi(Sys)$);
        Send response to RA;
        end !
    Else
    Load NewArchitecture;
    For each ($\tau_i$ in $\xi(Sys)$)
       $\xi_M$ (Sys)++;
       i ++;
    EndFor
    $\xi(Sys')$ = $\xi$ (Sys) + $\xi_M$ (Sys);
    $Risk_{aft}(\xi(Sys'))$;
    Output: Send Response (accepted or refused);
 EndFor
End
```

Finally, a response with a new set of tasks *Sys(t)* should be sent to *RA*. The proposed layer, presented in Algorithm 3, verifies the internet connection in order to capture the transmitted packets for each detected task.

Using the library "libpcap" in Android kernel, we perform a capture of the network traffic shown in Fig. 9. This capture is classified into two sets of tasks at t (*Sys*(t)) routed through the security agent to this layer are classified into two groups ($\xi_l(t)$ and $\xi_{nw}(t)$).



```
shell@android:/ # tcpdump -c 10
tcpdump -c 10
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
09:32:42.884941 IP 192.168.1.3.137 > 192.168.1.255.137: NBT UDP PACKET(137): QUE
RY; REQUEST; BROADCAST
09:32:43.396468 ARP, Request who-has 192.168.1.1 tell 192.168.1.4, length 28
09:32:43.601644 IP 192.168.1.3.137 > 192.168.1.255.137: NBT UDP PACKET(137): QUE
RY; REQUEST; BROADCAST
09:32:43.704731 IP 192.168.1.4.50177 > 239.255.255.250.1900: UDP, length 133
09:32:44.421206 IP 192.168.1.3.137 > 192.168.1.255.137: NBT UDP PACKET(137): QUE
RY; REQUEST; BROADCAST
```

Fig. 6. Information collection

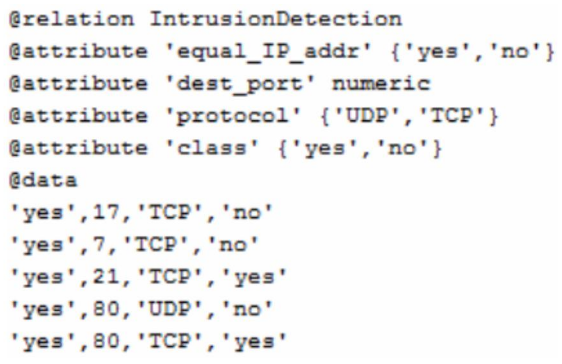
Algorithm 3 : Implementation of the Filter and detection layer

```
Begin
  1.Input: $\tau \leftarrow Tasks$;
  2.Foreach $(\tau)$
        If $connect_{wifi}()$;
            $capture_{packets}()$;
        End If
   3.$Collect_{informations}()$;
   4.Analysis();
   5.Switch($port_{dest}, addr_{src}, addr_{dest} and protocol$) do
        Detect();
        Filter();
        Output: Apply mechanisms;
   End
  End
End.
```

The captured packets are analyzed in order to obtain the necessary information Data: Source and destination IP addresses and also ports denoted by *An(t)=Data (CL(Sys(t)))*. These extracted information will be classified to prepare the intrusion detection *DI*(*t*). We have built a platform named Weka [5] exploiting the data mining algorithms. Then, we specify a file extension recognized by this platform to analyze it and determine the decision tree. This file must meet the form shown in Fig. 10. It gives rise to new rules that will build the basis of the decision tree, and we classify the result into two homogenous groups ("normal" and "abnormal"). Finally, the last level is the filtering. In this step, the system adds the adequate set of mechanisms Set_M with the appropriate parameters (T_j,C_j, D_j *and* SL_{M_j}) to the database *DBSA*. The value of security controls *SC* is also determined in order to evaluate the risk *RKr*(*aft*) of reconfiguration scenario Rg_k after adding the final solution. Our objective is to obtain *RKr*(*aft*) < *RKr*(*bef*).

The algorithm of the SchA is presented in Algorithm 4. It is based on two classes: (i) The Interpreter-based scheduling agent generates the OS tasks according to their security mechanisms and their parameters, and (ii) The generator-based agent scheduling is started by the "Interpreter" to calculate the parameters of each security mechanism $M_j(\alpha_j,\beta_j)$ defined in the work [4].

We developed a plugin at University of Carthage in Tunisia to allow secured reconfiguration scenarios of Android devices: (i) Evaluates the security of added-



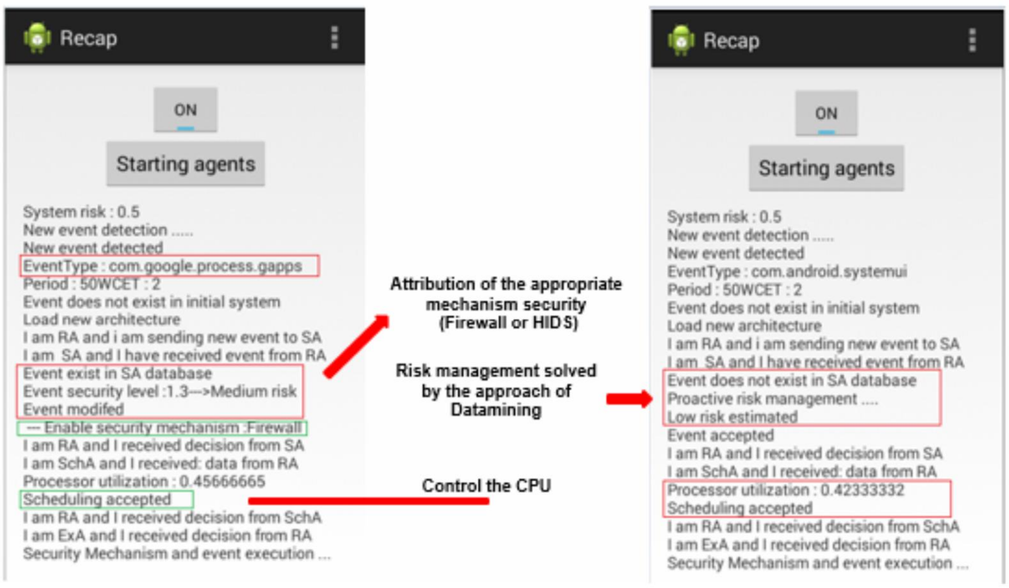
```
@relation IntrusionDetection
@attribute 'equal_IP_addr' {'yes','no'}
@attribute 'dest_port' numeric
@attribute 'protocol' {'UDP','TCP'}
@attribute 'class' {'yes','no'}
@data
'yes',17,'TCP','no'
'yes',7,'TCP','no'
'yes',21,'TCP','yes'
'yes',80,'UDP','no'
'yes',80,'TCP','yes'
```

Fig. 5. The file for the treatment of the decision tree

Algorithm 4 : Implementation of the Scheduling Agent

```
Begin
  1.Input : T (n Tasks), M (m mechanisms);
  2.i=1; j=1;
  3.While($i < n$ (Tasks)) do
     If (interpreter(T(i),M)=NULL) else execute(T(i));
     ElseIf (interpreter(T(i),M)=Mext)
       While ($j < n$(Mext)) do
         $\alpha$(j), $\beta$(j) $\leftarrow$ generate($\alpha$, $\beta$);
         If($\alpha$(j)=1) & ($\beta$(j)=1); then execute(M(j),T(j));
       EndWhile
     EndIF
   EndWhile
  3.Output: $\alpha$,$\beta$
End.
```

removed tasks, (ii) Evaluates the security level of the applied reconfiguration scenario after the addition-removal of tasks, and (iii) Allows or denies the system to execute this scenario. We show in Fig. 11 the detection of new tasks. The reconfiguration agent is started by verifying the existence of this new event in the system. There are two possibilities: (i) The task encodes the system before reconfiguration and therefore RA continues the execution of this task after verification of real-time contraints, or (ii) The system controls this new event by verifying the value of the security level and then a security solution will be generated. The security agent and the detection and filter layer are responsible to take the appropriate solution. When the task does not exist in the database, there are two possibilities to determine the risk of this task: (i) Applying the datamining or (ii) The sandboxing (Fig. 12).


Recap
ON
Starting agents
Attribution of the appropriate mechanism security (Firewall or HIDS)
Risk management solved by the approach of Datamining
Control the CPU
Recap
ON
Starting agents

```
System risk : 0.5
New event detection .....
New event detected
EventType : com.google.process.gapps
Period : 50WCET : 2
Event does not exist in initial system
Load new architecture
I am RA and i am sending new event to SA
I am  SA and I have received event from RA
Event exist in SA database
Event security level :1.3--->Medium risk
Event modifed
--- Enable security mechanism :Firewall
I am RA and I received decision from SA
I am SchA and I received: data from RA
Processor utilization : 0.45666665
Scheduling accepted
I am RA and I received decision from SchA
I am ExA and I received decision from RA
Security Mechanism and event execution ...
```

```
System risk : 0.5
New event detection .....
New event detected
EventType : com.android.systemui
Period : 50WCET : 2
Event does not exist in initial system
Load new architecture
I am RA and i am sending new event to SA
I am  SA and I have received event from RA
Event does not exist in SA database
Proactive risk management ....
Low risk estimated
Event accepted
I am RA and I received decision from SA
I am SchA and I received: data from RA
Processor utilization : 0.42333332
Scheduling accepted
I am RA and I received decision from SchA
I am ExA and I received decision from RA
Security Mechanism and event execution ...
```

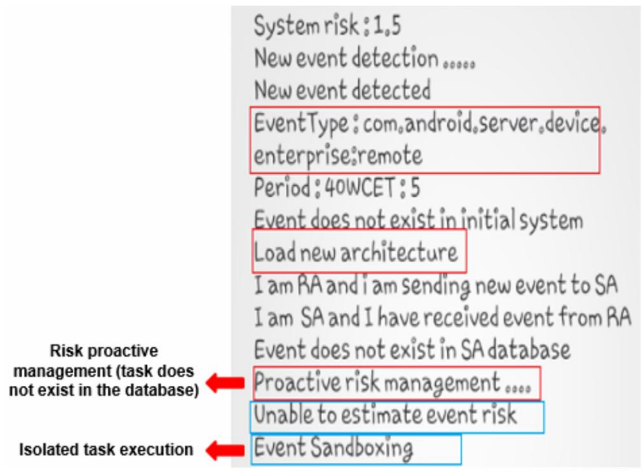
Fig. 7. Proposed solution: Added mechanisms

Table 2. Comparative study

Works	Architecture	Type of approach	Communication	Extracted features
[25]	Server architecture.	Static approach.	TCP\IP communication process High number of messages. High number of messages.	Apply cryptographic algorithms.
[34]	Android architecture.	Static approach.	Communication between components.	Apply the filtering method. Evaluate the overhead.
[39]	Client-Server architecture and mobile multi-agent architecture.	Static approach.	Communication with alerts signal.	Apply an agent-based intrusion detection system to detect suspicious behavior.
[41]-[42]	Android application.	Dynamic approach.	Event Triggering.	Use different detection and automatic exploration techniques.
[37]	Multi-agent architecture.	Static approach.	Message communication process.	Apply hybrid detection approaches.
[40]	Android application.	Static approach.	File share server.	Apply a filtering system using the APK file.
[20]	Service-oriented architecture (SOA).	Dynamic approach.	Communication between components.	Safe Building Security system based on security level.
Proposed Solution	Android and multi-agent Architecture.	Dynamic approach.	Message communication process.	Apply intrusion detection and filtering methods. Integrate them together. Safe Building Security system based on security level and real-time constraints.

5.3 Evaluation and comparitive study

To explain the result analysis of the proposed approach, we have implemented the prototype system on Android and analyzed the whole application as well as its status security in order to extract the OS tasks and their security levels. Then, we apply ten reconfiguration scenarios presented by adding-removing those software tasks (until arriving 40 tasks) in order to test and validate the proposed solution. The testing of the system's behavior before and after the application of the security agent is shown in Fig. 13. It is possible to note that the control of the security level *SL*



```
System risk : 1,5
New event detection .....
New event detected
EventType : com.android.server.device.
enterprise:remote
Period : 40WCET : 5
Event does not exist in initial system
Load new architecture
I am RA and i am sending new event to SA
I am SA and I have received event from RA
Event does not exist in SA database
Proactive risk management ....
Unable to estimate event risk
Event Sandboxing
```

Risk proactive management (task does not exist in the database)
Isolated task execution

Fig. 8. Applying sandboxing

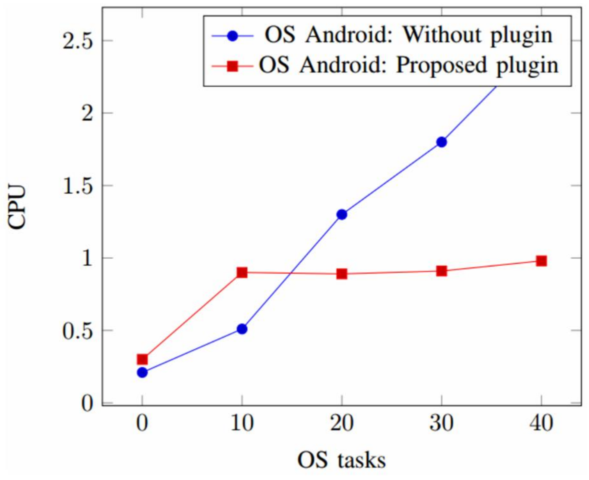


Fig. 14. CPU evaluation: Agent-based solution

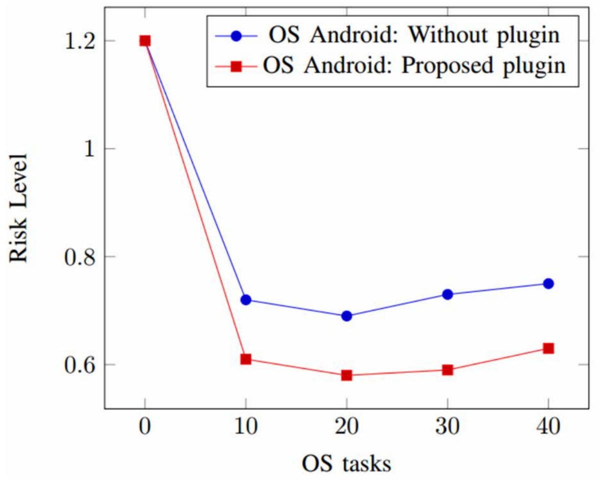


Fig. 13. Risk level evaluation: Agent-based solution

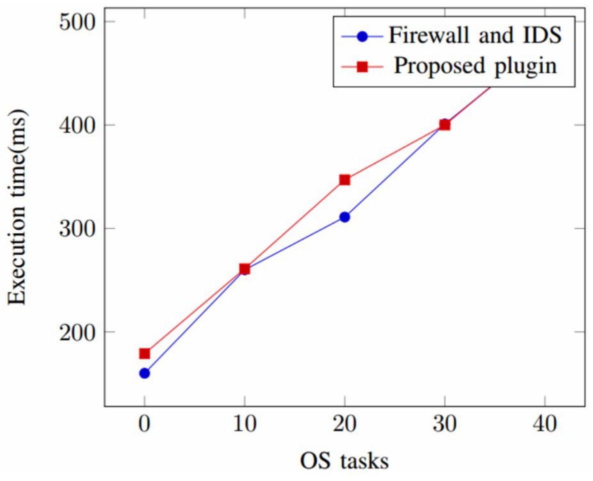


Fig. 15. Execution time evaluation

assists the system to proceed a correct execution. Indeed, according to *SLi* and *Rk* (formula (7)), we remark that the risk of the system without the intervention of the security agent is more than the risk related to the proposed system.

Table 2 describes a comparative table of the different researches. This table compares the developed approach in this paper with the related works in the literature ([42, 41], [37] and [40]) based on the used architecture, the approach type, the communication and the security techniques.

We notice that the proposed architecture makes the system more secure when a new reconfiguration form is applied. Contrariwise, when we do not apply the proposed solution, we observe that the system allows the violation of some security properties that may destroy the system behavior in some situations (Fig. 14).

The execution time is one of the significant parameters for the evaluation of the framework performance. This is why we use the time measurements based on the executed number of tasks. These measures have enabled us to plot the curve of Fig. 15. Besides, we present also another curve in this figure that shows the execution time of existing solutions of the IDS and Firewall in [34] and [41]. We notice that both approaches take almost the same execution time, but the proposed approach offers a higher security level as shown in Fig. 15.

After that, we run the proposed platform with these existing solutions by adding-removing some OS tasks with applying the ten reconfiguration scenarios proposed in Table 1. Those tasks are executed dynamically without the intervention of the user. The proposed solution described in Fig. 16 guarantees a minimum level of security.

6. Discussion and Conclusion

Although the applicability of the related works on reconfigurable and secured mobile systems is clear, we believe in their security limitations and also their classic solutions when reconfiguration scenarios are applied at run-time to add-remove-update software tasks for the system adaptation. In the proposed approach, the aim of the presented work is not to implement a new intrusion detection or filtering method, but rather to use an approach based on the existing intrusion detection and filtering in Android platform and to integrate them together by using the concept of adaptive agents. This combination allows us to correlate the different techniques in order to improve their effectiveness and to secure any automatic reconfiguration. Besides, this paper describes a new contribution to allow useful regulations of secured reconfiguration and improve the security mechanisms. The main goal is to implement secured software reconfigurations that adapt the mobile system to its environment while meeting security and real-time constraints. In fact, this idea is achieved by adding an autonomous plugin to be executed at run-time for checking security and temporal constraints. We have defined in this paper a plugin which is based on a middleware and an intelligent layer named the filtering and detection layer. The middleware is composed of four agents: reconfiguration agent which handles automatic reconfigurations, security agent which is defined to guarantee safe reconfigurations, scheduling agent which is defined to verify real-time constraints and execution agent which is responsible for the management of the system memory. The filtering and detection layer is based on four components: (i) Collection, (ii) Analysis, (iii) Detection and (iv) Filtering. The proposed approach reduces the execution of mechanisms or also the security level of tasks by overcoming the limitations of the Firewall and the IDS. We plan in our future work to study secured reconfigurable hardware components as well as networked distributed embedded systems. A real case study will be applied also for the evaluation of the proposed contributions.

Abbreviations and Glossary

CPU	: Central Processing Unit
IDS	: Intrusion Detection System
SL	: Security Level
OS	: Operating System
WCET	: Worst Case Execution Time
SC	: Security Controls
RA	: Reconfiguration Agent
SA	: Security Agent
ScA	: Scheduling Agent
Sys(*t*)	: Reconfigurable real-time system at time *t*
n_p	: Number of periodic tasks
n_{ap}	: Number of aperiodic tasks
n	: Number of tasks (n_p+n_{ap}) that can implement *Sys*
n'	: Number of tasks that can implement *Sys* at time *t*
n^{add}	: Number of added tasks to *Sys* at time *t*
n^{del}	: Number of removed tasks from *Sys* at time *t*
τ_i	: *i*-th task in *Sys* ($i \in [1..n]$)
D_i	: Deadline of task τ_i ($i \in [1..n]$)
C_i	: Computation time of task τ_i ($i \in [1..n]$)
T_i	: Period of task τ_i ($i \in [1..n]$)
λ_r^i	: Average arrival rate of an aperiodic task τ_i according to the poisson distribution
λ_c^i	: Average service rate of an aperiodic task τ_i according to the exponential arrival model
SL_{τ_i}	: Security level of task τ_i ($i \in [1..n]$)
δ_i	: Set of mechanisms attributed to τ_i ($i \in [1..n]$)
ξ_t	: Set of tasks that do not require a network connection
ξ_nw	: Set of tasks that require a network connection
Set_M	: Set of security mechanisms
n_m	: Number of security mechanisms
M_j	: *j*-th mechanism in Set_M ($j \in [1..n_m]$)
T_j	: Period of security mechanism M_j ($j \in [1...n_m]$)
SL_{M_j}	: Security level of mechanism M_j ($j \in [1...n_m]$)
β	: Number of occurrences of each mechanism M_j ($j \in [1...n_m]$)
α	: Scaling factor of period αT_j for M_j($j \in$ $[1...n_m]$)
Rg_k	: *k*-th reconfiguration scenario
Rk	: Risk of violating security constraints of tasks in *Sys*
Rk_j^r	: Risk of reconfiguration scenario Rg_j
Se_{HR}	: Threshold for a high security level
Se_{LR}	: Threshold for a low security level
U_{aper}	: Processor utilization of aperiodic tasks
U_{per}	: Processor utilization of periodic tasks
U_M	: Processor utilization of security mechanisms
U_{Sys}	: Total processor utilization U_{per} +U_{aper}+U_M
$U_{sys}^{bef}(t)$	: Processor utilization before reconfiguration
$U_{sys}^{aft}(t)$	: Processor utilization after reconfiguration
DFL	: Detection and filter layer based on the methodology of risk management
CL(*Sys*(*t*))	: Collection of information of *DFL*
Ant(*t*)	: Analysis of the collected information
DI(t)	: Detection intrusions by using data-mining

References

[1] J. F. Zhang, M. Khalgui, Z. Li, G. Frey, O. Mosbahi and H.B. Salah, "Reconfigurable coordination of distributed discrete event control systems," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 323-330, Jan. 2015.

[2] X. Wang, I. Khemaissia, M. Khalgui, Z. Li, O. Mosbahi and M. Zhou, "Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 258-271, Jan. 2015.

[3] R. Idriss, A. Loukil and M. Khalgui, "New solutions for feasible secured reconfiguration of embedded control systems," *In Proceedings of the 28th European Simulation and Modelling Conference ESM*, pp. 323-330, Jan. 2014.

[4] R. Idriss, A. Loukil and M. Khalgui, “New middleware for secured reconfigurable real-time systems,” *In Proceedings of the 14th International Conference on Intelligent Software Methodologies*, Tools and Techniques SoMeT, pp. 469-483, Sep. 2015.

[5] R. Robu and V. Stoicu-Tivadar, “Arff convertor tool for WEKA data mining software,” *In Proceedings of Computational Cybernetics and Technical Informatics ICCC-CONTI*, pp. 247-251, 2010.

[6] H. Dhouib, A. Loukil, A. Ammari and A. Jemai, “Proactive mobile agent security: A new access control approach based on the risk analysis,” *In Proceedings of the digital information processing and communications ICDIPC*, pp. 62-67, Aug. 2012.

[7] A. Shabtai, U. Kanonov and Y. Elovici, “Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method,” *Journal of Systems and Software*, vol. 83, no. 8, pp. 1524-1537, Aug. 2010.

[8] H. Khodor, S. Alaa, E. Imad, C. Ali and K. Ayman, “An implementation of intrusion detection system using genetic algorithm,” *International Journal of Network Security and Its Applications*, vol. 4, no. 2, pp. 109-120, Mar. 2012.

[9] C. Hsiao-Hwa and R.S. Hamid, “Security and communication networks,” *ISI Journal Citation Reports*, vol. 8, no. 2, pp. 54-77, 2015.

[10] C. Jindal, M. Chowkwale, R. Shethia and S.A. Shaikh, “A survey on intrusion detection systems for android smartphones,” *International Journal of Computer Science and Network*, vol. 3, no. 6, pp. 12-17, Oct. 2014.

[11] D. Fischer, B. Markscheffel and T. Seyffarth, “An overview of threats and security software solutions for smartphones,” *International Journal for Information Security Research*, vol. 3, no. 4, pp. 427-431, June. 2014.

[12] A. Oyeleye Christopher, Y. Daramola Comfort and A. James, “Mob-AIDS: An intrusion detection system for the android mobile enterprise,” *International Journal of the Digital Information Processing and Communications*, vol. 11, no. 3, pp. 161-167, May. 2014.

[13] W. Jiang, K. Jiang, X. Zhang and Y. Ma, “Energy optimization of security-critical real-time applications with guaranteed security protection,” *Journal of Systems Architecture,* vol. 61, no. 7, pp. 282-292, Aug. 2015.

[14] W. Wang, R. Sanjay and M. Prabhat, “Energy-aware dynamic reconfiguration algorithms for real-Time multitasking systems,” *Elsevier Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 35-45, Mar. 2011.

[15] W. Li, “Evaluating the impacts of dynamic reconfiguration on the QoS of running systems,” *Journal of Systems and Software*, vol. 84, no. 12, pp. 2123-2138, Dec. 2011.

[16] Z. Zhao, P. Zhao and W. Li, “Quantitative analysis of design decisions for dynamic reconfiguration,” *Journal of Software*, vol. 8, no. 10, pp. 2391-2396, Oct. 2013.

[17] B. Pascal, T. Lionel, S. Gilles, R. Michel, and S. Nicolas, “Run time mapping for dynamic reconfiguration management in embedded systems,” *International Journal of Embedded Systems*, vol. 4, no. 3-4, pp. 276-291, 2010.

[18] K. Vallidevi and B. Chitra, “SLA aware dynamic reconfiguration for the safe building security system,” *International Journal of Software Engineering and Its Applications*, vol. 9, no. 2, pp. 151-162, Feb. 2015.

[19] F. Zhao, J. Wang, Ju. Wang and J. Jonrinaldi, “A dynamic rescheduling model with multi-agent system and its solution method,” *Journal of Mechanical Engineering*, vol. 58, no. 2, pp. 81-92, 2012.

[20] A. Hussein Rady, “Multi-agent system for negotiation in a collaborative supply chain management,” *International Journal of Video Image Processing and Network Security IJVIPNS-IJENS*, vol. 11, no. 5, pp.25-36, 2011.

[21] W. Jiang, Z. Guo, Y. Ma and N. Sang, “Measurement-based research on cryptographic algorithms for embedded real-time systems,” *Journal of Systems Architecture JSA*, vol. 59, no. 10-Part D, pp. 1394-1404, Nov. 2013.

[22] Y. Verbelen, A. Braeken, S. Kubera, A. Touhafi, J. Vliegen and N. Mentens, “Implementation of a server architecture for secure reconfiguration of embedded systems,” *ARPN Journal of systems and software*, vol. 1, no. 9, pp. 270-279, Dec. 2011.

[23] T. Thanh, T. Hoang Vu, N. Duy Phuong, D. Son Tung, C. Nguyen-Van, N. Van Cuong and P. Ngoc Nam, “Enhance performance in implementing the security of partially reconfigurable embedded systems,” *American Journal of Embedded Systems and Applications*, vol. 2, no. 1, pp. 1-5, Jan. 2014.

[24] S. Lee and D.Y. Ju, “A novel method to avoid malicious applications on android,” *International Journal of Security and Its Applications*, vol. 7, no. 5, pp. 121-130, Sep. 2013.

[25] C-H. Huang, P-A. Hsiung and J-S. Shen, “UML-based hardware/software co-design platform for dynamically partially reconfigurable network security systems,” *Journal of Systems Architecture JSA*, vol. 56, no. 2-3, pp. 88-102, 2010.

[26] K. Kularbphrttong, S. Somngam, C. Tongsiri and P. Roonrakwit, “A recommender system using collaborative filtering and K-Mean based on android application,” *In Proceedings of International Conference Applied Mathematics, Computational Science and Engineering*, pp. 161-166, Dec. 2014.

[27] P. Gayathri and A. Rama, “Smart shop search android mobile application,” *International Journal of Innova-*

tive Research in Science, Engineering and Technology, vol. 4, no. 2, pp. 146-150, Feb. 2015.

[28] C. Kotkar Chetan and P. Game, "Exploring security mechanisms to android device," *International Journal of Advanced Computer Research*, vol. 3, no. 13, pp. 216-221, Dec. 2013.

[29] Q. Qian, J. Cai, M. Xie and R. Zhang, "Malicious Behavior Analysis for Android Applications," *International Journal of Network Security*, vol. 18, no. 1, pp. 182-192, Jan. 2016.

[30] T. Hayajneh, B.J. Mohd, A. Itradat and A.N. Quttoum, "Performance and information security evaluation with firewalls," *International Journal of Security and Its Applications*, vol. 7, no. 6, pp. 355-372, 2013.

[31] A. Magdy, M. Mahros and E. Hemayed, "Firewall-based solution for preventing privilege escalation *attacks in android," International Journal of Computer Networks and Communications Security*, vol. 2, no. 9, pp. 318-327, Sep. 2014.

[32] L. Dua and D. Bansal, "Review on mobile threats and detection techniques," *International Journal of Distributed and Parallel Systems*, vol. 5, no. 4, pp. 21-29, July. 2014.

[33] P. Jae-Kyung and C. Sang-Yong, "Studying security weaknesses of android system," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 7-12, Mar. 2015.

[34] A. J. Alzahrani and A. A. Ghorbani, "A multi-agent system for smartphone intrusion detection framework," *In Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, vol. 1, pp. 101-113, 2015.

[35] M. Alam, and S. T. Vuong, "An intelligent multi-agent based detection framework for classification of android malware," *In Proceedings of the 10th International Conference of Active Media Technology*, vol. 8610, pp. 226-237, 2014.

[36] A. Dwived, Y.K. Rana and B.P. patel, "A Real time host and network mobile agent based intrusion detection system (HNMAIDS)," *International Journal of Computer Applications*, vol. 113, no. 12, pp.33-40, 2015.

[37] M. Jang, and D. Kim, "Filtering illegal Android application based on feature information," *In Proceedings of Research in Adaptive and Convergent Systems*, pp.357-358, 2013.

[38] V. Rastogi, Y. Chen and W. Enck, "AppsPlayground: automatic security analysis of smartphone applications," *In Proceedings of the third ACM conference on Data and application security and privacy*, pp. 209-220, 2013.

[39] L. Min and Q. Cao, "Runtime-based behavior dynamic analysis system for Android malware detection," *Advanced Materials Research*, vol. 756-759, pp. 2220-2225, 2012.

[40] T. Blasing, L. Batyuk, A.D. Schimdt, S.H. Camtepe and S. Albayrak, "An Android application sandbox system for suspicious software detection," *In Proc. the 5th International Conference on Malicious and Unwanted Software*, DOI: 10.1109/ MALWARE. 2010.5665792, 2010.

[41] H.C. Liu, J.X. You, Z. Li and G. Tian, " Fuzzy Petri nets for knowledge representation and reasoning: A literature review," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 45-56, 2017.

[42] F. Yang, N. Wu, Y. Qiao, M. Zhou and Z. Li, "Scheduling of singlearm cluster tools for an atomic layer deposition process with residency time constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 502-516, 2017.

[43] S.B. Meskina, N. Doggaz, M. Khalgui and Z. Li, "Multi-agent framework for smart grids recovery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. DOI: 10.1109/TSMC.2016.2573824, 2016.

[44] M. Gasmi, O. Mosbahi, M. Khalgui, L. Gomes and Z. Li, "RNode: New pipelined approach for an effective reconfigurable wireless sensor node," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2016.2625817, 2016.

[45] H. Grichi, O. Mosbahi, M. Khalgui and Z. Li, "RWiN: New methodology for the development of reconfigurable WSN," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 109-125, 2017.

[46] H. Grichi, O. Mosbahi, M. Khalgui and Z. Li, "New poweroriented methodology for dynamic resizing and mobility of reconfigurable wireless sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol.99, pp. 1-11, 2017.

[47] M.O. Ben Salem, O. Mosbahi, M. Khalgui, Z. Jlalia, G. Frey and M. Smida, "BROMETH: Methodology to design safe reconfigurable medical robotic systems," *International Journal of Medical Robotics and Computer Assisted Surgery*, DOI: 10.1002/rcs. 1786, 2016.

[48] M. Uzam, Z. Li, G. Gelen, and R. S. Zakariyya, "A divide-andconquer-method for the synthesis of liveness enforcing supervisors for flexible manufacturing systems," *Journal of Intelligent Manufacturing*, vol.27, no. 5, pp. 1111-1129, 2016.

[49] Y.F Chen, Z. Li, A. Al-Ahmari, N.Q. Wu and T. Qu, "Deadlock recovery for flexible manufacturing systems modeled with Petri nets," *Information Sciences*, Vol.381, pp. 290-303, 2017.

[50] Y. Tong, Z. Li and A. Giua, "On the equivalence of observation structures for Petri net generators," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2448- 2462, 2016.

[51] Y. Tong, Z.W. Li, C. Seatzu and A. Giua, "Verification of state-based opacity using Petri nets,"

IEEE Trans. on Automatic Control, vol. 62, no. 6, pp. 2823-2837, 2017.

[52] X. Wang, Z. Li and W.M. Wonham, "Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 1, pp. 101-111, 2016.

Rim Idriss was born in Tunis in 1988. She obtained the engineer degree in Computer Networks and Telecommunications (RT) at the National Institute of Applied Sciences and Technology (INSAT) in 2013. She is a PhD student in computer science at the University of Carthage (INSAT). She has joined the computer lab for industrial systems (LISI) in 2014. She has a comprehensive understanding of issues related to security of network and systems architecture and technical knowledge for their implementation.

Adlen LOUKIL M.A., Ph.D., is Associate Professor at Tunis National Institute of Applied Sciences (INSAT), which he joined in 1997. Dr. LOUKIL received his Ph.D. in Computer Science from National Institute of Applied Sciences INSA-Lyon (France). Dr. LOUKIL has co-authored and/or edited three books along with more than 30 peer-reviewed articles and reviews. He teaches courses and conducts research in the areas of information security, cloud computing and risk management. His main area of research is proactive security, where the goal is to develop cyber security protection methods that integrate risks from multiple independent sources.

Mohamed Khalgui received the B.S. degree in Computer Science from Tunis El Manar University, Tunis, Tunisia, in 2001, the M.S. degree in Telecommunication and Services from Henri Poincare University, Nancy, France, in 2003, the Ph.D. degree from the National Polytechnic Institute of Lorraine, Nancy, France, in 2007, and the Habilitation Diploma degree in Computer Science from Martin Luther University of HalleWittenberg, Halle, Germany, in 2012, with Humboldt Grant. He was a Researcher in Computer Science with the Institut National de Recherche en Informatique et Automatique INRIA, France, ITIA-CNR Institute, Italy, Systems Control Laboratory, Xidian University, Xian, China, KACST Institute in KSA, Riyadh, Saudi Arabia, a Collaborator with SEG Research Group, Patras University, Patras, Greece, Director of RECS Project at O3NEIDA, Canada, Director of RES Project at Synesis Consortium, Italy, Manager of Cyna-RCS Project at Cynapsys Consortium in France, and Manager of BROS and RWiN Projects at ARDIA Corporation in Germany. He has been involved in various international projects and collaborations. Dr. Khalgui is a member of many TPC of conferences and different boards of journals.

Zhiwu Li (M06S M07F16) received the B.S. degree in mechanical engineering, the M.S. degree in automatic control, and the Ph.D. degree in manufacturing engineering from Xidian University, Xian, China, in 1989, 1992, and 1995, respectively. He joined Xidian University in 1992. His current research interests include Petri net theory and application, supervisory control of discrete event systems, workflow modeling and analysis, system reconfiguration, game theory, and data and process mining. He is currently with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, China. Dr. Li was a recipient of the Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation and Germany. He is listed in Marquis Whos Who in the World, 27th edition, 2010. He is the Founding Chair of the Xian Chapter of the IEEE Systems, Man,and Cybernetics Society.

Abdulrahman Al-Ahmari, dean of advanced manufacturing institute, Supervisor of Raytheon Chair for Systems Engineering, and professor of Industrial Engineering at King Saud University, Riyadh, Saudi Arabia. He received a Ph.D. in Manufacturing Systems Engineering from the University of Sheffield in 1998. Professor Al-Ahmari has published papers in leading journals of Industrial and Manufacturing Engineering. He led a number of funded projects from different organizations in Saudi Arabia. His research interests are advanced manufacturing technologies, Petri nets, analysis and design of manufacturing systems; Computer Integrated Manufacturing (CIM); optimization of manufacturing operations; FMS and cellular manufacturing systems and applications of DSS in manufacturing.