Analog Integrated Circuits and Signal Processing
https://doi.org/10.1007/s10470-018-1167-8

A framework for enhancing mobile workflow execution through injection of flexible security controls

Borja Bordel[1] · Ramón Alcarria[1] · Augusto Morales[2] · Ignacio Castillo[3]

Received: 2 November 2017 / Revised: 5 March 2018 / Accepted: 15 March 2018

Abstract
Mobile workflow execution is gaining importance as traditional process execution systems are employed in many new scenarios such as mobile networks or the Internet of Things. Unfortunately, in these solutions, security is still based on control loops or computer science techniques which have not evolved as fast as current mobile systems and applications. In this context, in order to improve the security level of these systems, it is necessary to create a security framework tightly coupled with the mobile workflow execution platforms. To contribute filling this gap, we propose a framework to inject security controls in workflows, which supports mobile execution and allows a flexible decision making. This solution models security as control points where some relevant previously defined indicators are evaluated. Depending on the obtained values, the framework takes corrective, preventive or adaptive actions, considering also the execution system capabilities and the workflow being executed. In order to evaluate the effectiveness and performance of the proposed solution we include experimental validation.

Keywords Mobile workflow execution · Security modeling · Security controls · Security injection

1 Introduction

Mobile devices are a key part of daily life nowadays. People employ them for a wide range of activities such as surfing to the Internet, making calls (using Voice-over-IP -VoIP- technologies), and even monitoring in real time locations, biological signals and other people. Furthermore, as mobile device capabilities increases, industrial companies are also integrating mobile devices into their production systems, following the principles of the Industry 4.0 paradigm [1].

Most of the described scenarios require the definition of workflows (in an implicit or explicit way) whose execution, obviously, must support mobility. Implicit workflows are triggered by the mobile applications running into devices when a certain action is detected (for example, opening a web browser). On the other hand, explicit workflows are defined by people and the workflow execution system is explicitly requested to execute them in a certain moment. Moreover, most of these activities (workflows) require communicating with Internet-of-Things (IoT) devices, third-party cell phones, Cyber-Physical devices and production systems [2] and/or cloud-based servers which store and forward personal data and provide tailored services.

Therefore, mobile devices turn into critical components when workflows are executed. In this context, mobile workflows require techniques to avoid fails during executions (e.g. dead-locks), to detect risky situations derived from the mobile nature of devices (coverage problems, battery failures, etc.), to implement an adequate management policy of personal information, and/or to guarantee a

✉ Augusto Morales
moralesaugusto@ieee.org

Borja Bordel
bbordel@dit.upm.es

Ramón Alcarria
ralcarria@dit.upm.es

Ignacio Castillo
icastillo@ieee.org

[1] Technical University of Madrid, Madrid, Spain
[2] Check Point Software Technologies, Mexico City, Mexico
[3] Autonomous University of Mexico, Mexico City, Mexico

safe communication among components and devices (among other needs).

In fact, core security requirements, such as the previously cited confidentiality, integrity or availability, are becoming a foremost concern in today's life [3]. However, despite the amount of the components that could came across the execution of mobile workflows [4], these requirements are nowadays still addressed with computer science solutions (as controls in the network perimeter layer, e.g. with firewalls and VPNs; in the OS layer e.g. sandboxing, or at the application layer e.g. with antivirus), or with traditional industrial techniques [5].

Thus, there is a major challenge related to how to adapt these controls to the high dynamism and decoupled execution of mobile workflows and their computing environment. Besides, new security solutions specifically designed to address the new risk of the world wide connected society should be also proposed (for example, tourists who are monitored by their medical center in their country of origin thousands of miles away).

One of the most effective manners of securing workflows and, in general, execution systems is the injection of security controls along the execution model (workflow) [6]. Security controls are evaluation points where some relevant previously defined indicators are measured. This set of indicators must be proposed in such way that they represent the security state of the workflow execution. Relevant indicators strongly depend on the application scenario; any case, security controls are countermeasures that mitigate the probability of a threat affecting a workflow execution independently of their fixed or mobile nature. Moreover, as a mobile workflow could make use of a variety of local, remote or cloud-based systems to carry out its purposes, the security controls shall be adjusted to the entire workflow lifecycle [7], from its creation to its end. This fact is especially critical during runtime, when a mobile workflow must deal with a set of constraints such as connectivity, power processing, network heterogeneity and user experience [4].

Therefore, we contribute to a security framework that injects security controls in workflows. These controls are evaluated through a special infrastructure belonging to the data plane. The results obtained in the injected security controls allow taking decisions in a flexible way, depending on the execution system capabilities and the workflow context. Therefore, the overall mobile execution is enhanced in such way that confidentiality, integrity and availability principles are met.

Although, as said, decisions may be taken in a flexible way, three different types of actions are identified: preventive, corrective and adaptive. We also describe: (1) the required transactions to support the security control injection, (2) the required techniques to evaluate these control points in a decoupled way from the rest of the execution system (in that way the proposed solution is valid for every proposed workflow execution system) and (3) the necessary interactions between the different implemented components depending on the type of actions to be taken.

The paper structure is as follows: Sect. 2 reviews related works addressing the problem of security in distributed workflow execution. Section 3 presents the three different action types and some example scenarios. Section 4 addresses in detail the flexible mobile execution, describes the overall model and messaging flow. Finally, Sects. 5 and 6 describe the experimental validation, present the obtained results and conclude the paper.

2 Related work

Several works have addressed the problem of security in distributed workflow execution. Some authors [8] propose algorithms for workflow fragmentation in federated clouds [9], which present complex security features that are addressed by the provision of a security model based on rules. These security models have been formalized by other authors such as Chang et al. [10], which investigated a cloud computing security framework, based on the development and integration of three major security technologies: firewall, identity management, and encryption based on the development of enterprise file sync and share technologies.

Workflow fragment distribution in cloud environments has previously been addressed by the authors of this paper [4], although the proposed cost model did not contain specific rules for security provision, which is provided in this work. Other authors [11] seek to reconcile efficient scheduling and allocation of workflows in hybrid clouds and the fulfillment of security requirements, measuring the number of mutually untrusted tenants assigned to the same virtual infrastructure. Security, in this kind of works is related with the set of mechanisms that allow a robust service management and reliable execution. There are studies that address security issues in workflow execution from the point of view of protection of workflow information. In the case of [12] data confidentiality and integrity are addressed by encrypting information generated by workflow tasks, determining which communication between tasks must be encrypted and which, for performance reasons, may be sent without encryption. All these works only consider cloud environments (federated or hybrid), and their contributions address the deployment process.

Other works such as [13] propose solutions to secure workflow enactment while providing an identification services to the different workflows, however, do not tackle the delegation of these services as part of mobile environment were non-stable connections could hamper the workflow execution. Some authors [14] tackle the problem of decentralized workflow by focusing on traceability and control access policies, while assuming that other critical security controls are resolved at the orchestrator level.

In the field of mobile workflows, important aspects have been identified in workflow and service composition security, constraints of mobile environments and the requirements to execute a business process or workflow in mobile domains [15]. These aspects have been taken into account in implementing solutions such as T-RBAC [16], an access control model that assigns permissions to tasks instead of roles. It uses the workflow authorization model for synchronizing workflow with authorization flow.

In the field of the definition of task allocations schemes, there are solutions that use dependent security models to decide the best execution strategy depending on contextual information or distributed regions. In the case of [17] tasks allocation schemes based on energy and scalability are provided, which is a similar interest of our research. In [18] a trade-off fault-tolerance mechanism based on genetic algorithms for workflow offloading to cloud servers is proposed. However, in both works, the problem of modeling security controls within the tasks execution environment is not addressed. Abrishami et al. [19] propose a new quality-of-service (QoS) based workflow scheduling algorithm by the partial critical path (PCP) technique in grid environment, which tried to minimize the cost of workflow execution with the deadline constraint. The work of Zeng et al. [20], very related to the previous one, introduce a security-aware workflow scheduling strategy, which proposed an economical distribution of tasks among the available cloud security services. Our contribution, however, goes further, proposing a flexible way to minimize the workflow execution cost under security constraints.

Finally, in the field of scientific workflows, security is increasingly critical for complex data applications to be executed on large-scale distributed infrastructures. Li et al. [21] propose a security and cost aware scheduling algorithm, based on particle swarm optimization, for heterogeneous tasks of scientific workflow in clouds. Tang et al. [22] designed a security-driven scheduling architecture to measure the trust level of each node and to estimate task security overhead with priority ranks. However, these algorithms only apply for geographically distributed computing environments, where other considerations such as device's and server's technical requirements should be considered.

As conclusions of the study, state-of-the-art solutions do not properly address a flexible security control in mobile workflow executions, taking into consideration the current mobility requirements, the possibility of workflow alterations, and the characteristics and security risks of the executing environment. Therefore, we propose a control-based security framework to provide flexible security characteristics in mobile workflow execution, in a preventive, corrective and detective way.

3 Security actions: motivating scenarios

To clarify our vision related how modeling flexible security controls can enhance mobile workflow execution, in this section the three different security action types (preventive, corrective and adaptive) previously identified are formally proposed and described. Besides, for each action type, a first motivation scenario is described.

When the execution system reaches a control point, the security control is performed, and the defined security indicators are evaluated. Then, preventive security actions are performed to prevent the workflow's alteration or non-authorized modifications from happening. Corrective actions repair the workflow execution. And adaptive actions adapt the entire execution to the underlying context characteristics in order to improve the efficiency and reduce the risk of future workflow's abnormal deviation or possible attacks.

3.1 Preventive actions

Preventive actions are actions performed in order to avoid an imminent execution fail or the effects of an unavoidable risk in the near future. Security controls, in order to trigger preventive actions, must obtain undeniable evidences of the imminent and unavoidable appearance of an execution fail.

These actions, thus, answer predictable fails such as battery discharges, exits from the coverage area, etc.

One relevant scenario where these actions may appear is the access to e-commerce platforms using mobile devices.

Bob (i.e. a certain local mobile agent) is travelling from an airport and running out of battery. Bob executes an application which underneath has a workflow that makes a bank account operation, verifies his hotel reservation, flight connections and payments, while using the public airport Wi-Fi. His cellphone moves to a battery save mode and disables identification components such as fingerprint and location sensors. As authentication and authorization must be maintained preventive security controls arise, therefore, the workflow transfers identification and authorization services to the cloud and establishes a secure connection. This new virtual service allows the workflow to be seamlessly executed as it fetches the location and anti-fraud controls required by his bank.

From the point of view of the interaction flow (Fig. 1), the preventive scenario allows the security framework to

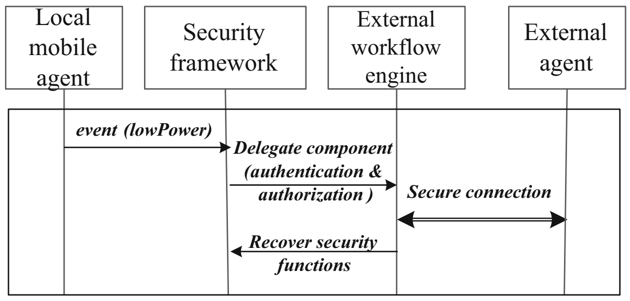


Fig. 1 Interaction flow of security controls in preventive scenario

detect contextual information which can trigger a change in security policy. In this scenario, a technical requirement of one of the devices (turns into battery saving mode) produces an event (*lowPower*) that is controlled by the security framework, which will trigger a preventive action, delegating the authentication and authorization components to an external execution engine, which will establish the proper secure connections with external services such as banking server.

3.2 Corrective actions

Corrective actions are actions performed to mitigate the effects of an execution fail that has already taken place, remove the malicious behavior of an attack that is already done, or to grant permissions to a component that is no longer suspicious. Corrective actions are not designed to avoid risks, but to redirect the execution of a workflow that already has problems (these problems are which are detected by the security controls).

These actions are probably the most generic, so many relevant scenarios could be defined as the following. Bob (i.e. a certain local mobile agent) is a senior citizen who needs constant monitoring, so his smartwatch runs a workflow that monitors vital signs, collects motion sensors data and analyzes this information to detect emergencies. Bob installs one component (workflow fragment) that solicits the subscription of Bob's personal information to send it to emergency services in case there is an emergency. As this new component solicits access to personal data, security controls must detect if this is a malicious component or it is trustworthy. In this last case some corrective actions must be performed by the security framework to securely delegate the execution of a workflow fragment to an external execution engine. The credentials of the smartwatch, as temporal encryption keys, are sent to the emergency services (e.g. ambulance and health workers), which also run in their mobile device an instance of the Bob's workflow, allowing them to uniquely identify Bob, locate his health information and to carry out the exact medical procedure. As his health data is protected by regulations, it will only be available when Bob's workflow triggers the temporal encryption keys.

Regarding the interaction flow (Fig. 2), the corrective scenario describes how the security framework obtains the necessary permissions from local mobile agents to delegate the execution of a workflow fragment to an external execution engine. This external workflow engine, by possessing temporary credentials for device access, can establish a secure connection without the personal information of local mobile agent being compromised.

3.3 Adaptive actions

Adaptive actions are actions performed to improve the efficiency of workflow execution and to reduce the probability of suffering an execution fail in the future or the effects of a certain risk, which is only potentially dangerous in an uncertain future. Adaptive actions are performed if the security control detects a change in the underlying conditions of the hardware platform or the execution context. These changes are not dangerous in the near future nor do they imply an imminent failure, but increase the probabilities of problems in the future, so it is decided to adapt the execution to the new conditions.

These actions are, probably, the less common ones, but some application scenarios may be also defined. One possible scenario is an enforcement agency communication. Alice (Local mobile agent A) and Bob (Local mobile agent B) are two law enforcement agents participating in an undercover operation. In their mobile device they execute a distributed workflow. It timely turns on the microphone on Alice's device, gets her location data, hijack existing Bluetooth devices and finally use Bob's device as a gateway to send a message with the collected information. As the workflow detects that is not possible to create a direct wireless connection between their devices, because they are in a hostile environment (e.g. SSL interception, promiscuous cards), they change the running state to non-

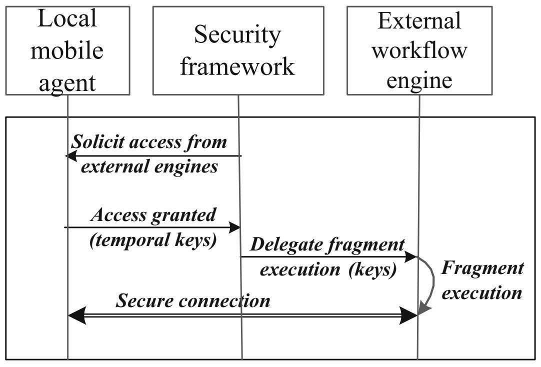

Local mobile agent
Security framework
External workflow engine
Solicit access from external engines
Access granted (temporal keys)
Delegate fragment execution (keys)
Fragment execution
Secure connection

Fig. 2 Interaction flow of security controls in corrective scenario

secure so both enable detective controls to detect and correct *MiTM* intents.

Considering the interaction flow presented in Fig. 3, the adaptive scenario reflects how the security framework continuously checks the status of executing workflows. Thus, once a fault in the execution of a task is detected (Activity execution failed), the security framework can activate some secondary branch related to security, such as the notification to the Local mobile agents A and B that are in an environment where security may have been compromised and they must configure their communication mode accordingly.

4 Flexible mobile workflow execution

The workflow model employed in this paper (based on tasks and activities) evolves from the previously presented proposals about this topic [4]. A workflow (see Fig. 4) is composed of a set of tasks, which may be decomposed into activities.

A Task (Ts) is a workflow fragment, which is instantiated in order to execute some activities in the mobile device. Tasks are arranged and initialized in the workflow initialization process. An Activity (At) is an atomic unit of a task, and it represents the communication with resources that can be either location info via sensors' readings or request of an internal database key and value (e.g. an http cookie/credentials).

On the other hand, each mobile device has a local orchestrator (execution engine) that analyzes the entire workflow, organizes the execution order of the different tasks and coordinates the execution of each workflow fragment (Task). Besides, we assume that an existing middleware will be capable of supporting the activity execution with local and remote resources in data plane. Hence, our overall model is abstracted from the underlying OS capabilities and independent of the invocation of OS

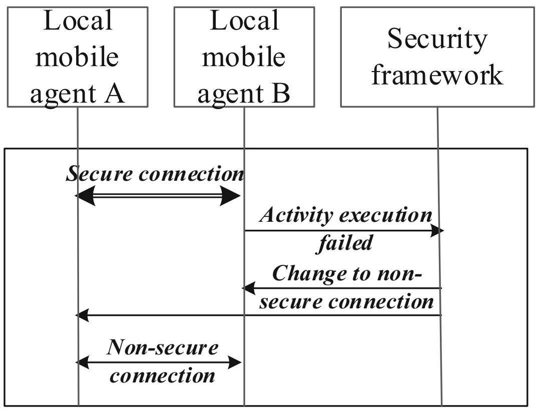


Fig. 3 Interaction flow of security controls in adaptive scenario

APIs in data plane. In the proposed workflow and execution models, furthermore, each activity composing a task communicates with each other via *iNotify()* primitives (see Fig. 5). This generic primitive describes the state of each activity (*ready*, *blocked*, *running* and *done*), allowing the deployed security framework to keep track of the execution flow.

The model supports activities (selected by the users of by the workflow engine during initialization process) to inject some security controls are injected as evaluation points. This points includes an invocation to the *eNotify()* primitive.

eNotify is a primitive that triggers a Control Event (*Cevt*), which is received by Security Activity which executed a *Receive()* primitive. These Control events provide Interprocess Communications (IPC) for different threads/processes, decoupling the security framework from the execution system. In fact, the reason of using the *eNotify()* primitive is to decouple in time and space the variety of activities running from the Security Activity and enable Inter Process Communication capabilities.

The cited "Security Task" (SA) is a virtual task belonging to the proposed security framework (see Fig. 5) which provides a uniform interface, similar to which provides every standard activity. In that way, the triggered Control Event is received by the Security Task as any other activity could receive a regular event.

In order to allow the execution of security actions, the triggered Control event identifies the specific instance that triggered it, as well as the location of that entity: *LOCAL* or *REMOTE*. It also classifies the current security level in the execution system: *LOW, MEDIUM, HIGH*.

The Security Task evaluates the security indicators using the appropriate primitives (performs the security control) and depending on the obtained results, and the information provided by the Control Event, selects the security action (or actions) to be taken: *PREVENTIVE, CORRECTIVE,* and/or *ADAPTIVE*. These actions may be directly invoked by the SA using the adequate API offered by the operating systems (OS), if the workflow execution can continue without changes. On the other hand, sometimes, the selected actions should modify the workflow execution in order to improve the security level, so the SA invokes the *setatype()* primitive in order to inform the corresponding orchestrator about the new situation. In order to maintain the security framework and the execution system decoupled, this primitive triggers a "Security Event" *Sevt*, which is received by the orchestrator by means of the *receive()* primitive.

For example, in the scenario described in Sect. 3.1, the security activity detects the low battery level and encloses the need of a preventive security control. In the scenario described in Sect. 3.2 the activity detects an abnormal level

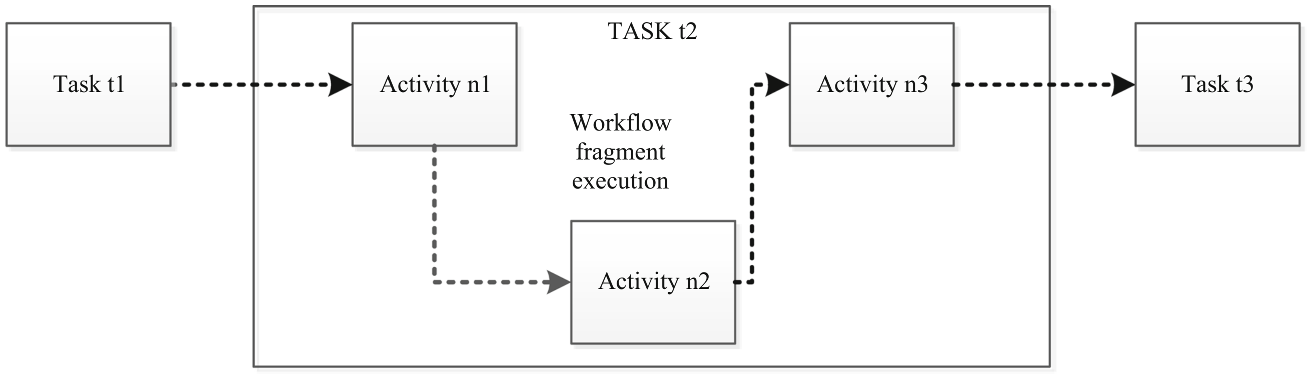


Fig. 4 Workflow model

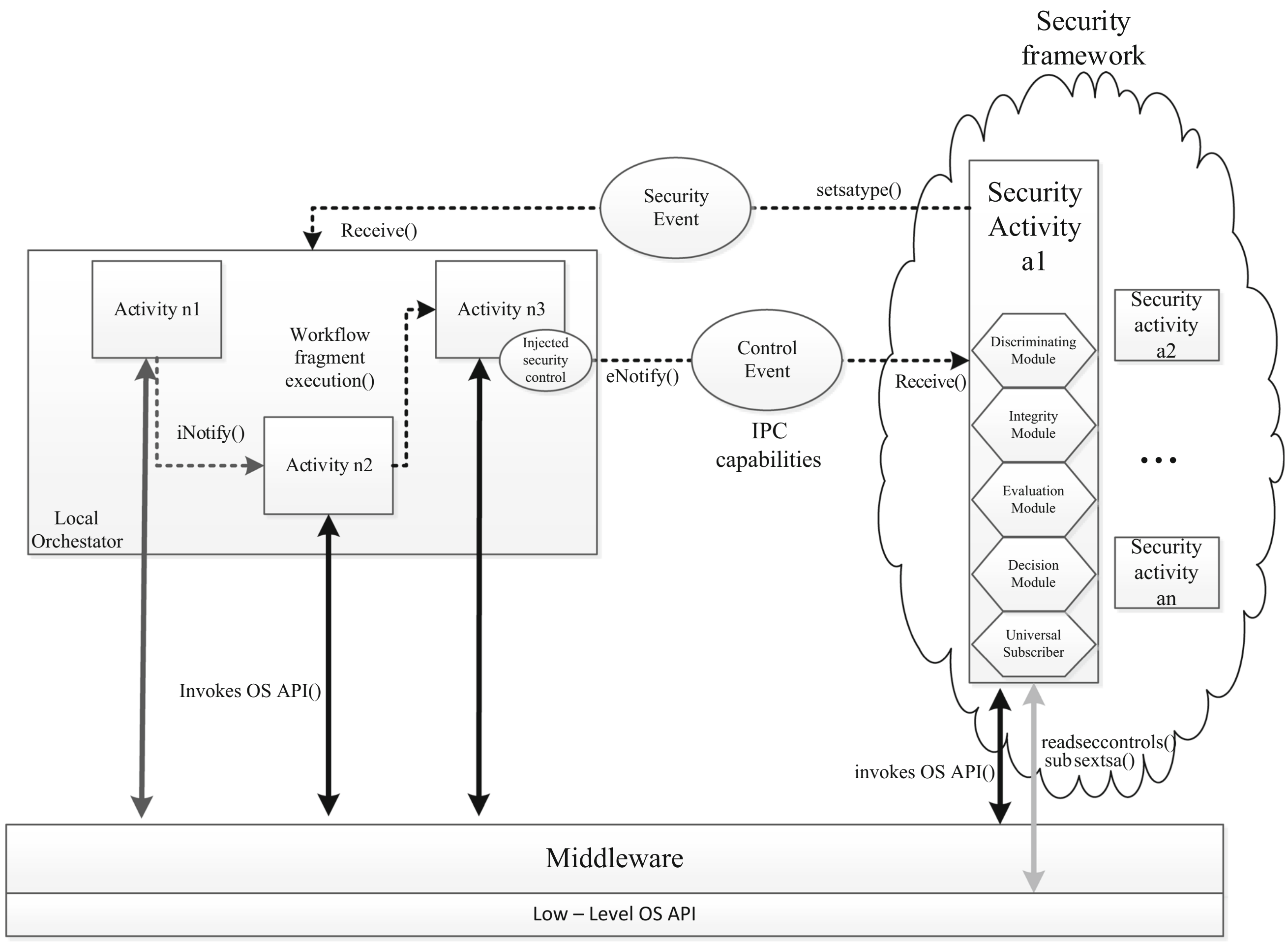


Fig. 5 Proposed Security framework

of blood pressure and requests a corrective action. The next subsections describe each one of the components shows in Fig. 5 in more detail.

4.1 Security activity

The Security Activity (SA) performs the security controls (evaluates the relevant security indicators) using the readseccontrols*()* primitive, which extracts the required information from the underlying platform. In this process, *readseccontrols()* makes usage of Low-Level OS APIs (as, probably, the deployed middleware cannot access to the needed information), so taking as example the current two most popular mobile OS: Android and iOS, it would be necessary to use special runtime permissions and private APIs respectively.

On the other hand, as said, some remoted entities could also trigger security events. In that way, when the SA receives a *Cevt*, it processes the received content (information) about the location of the original activity, and activates the security control locally or remotely. Depending on the obtained results, the SA may also order some security actions to be performed locally or remotely.

In order to be able to extract in an easy way the needed information from the received control events, every *Cevt* is described using an XML document. Figure 6 shows an example of the proposed structured for a *Cevt*.

To make the messaging aligned to low-processing requirements of mobile workflows, we enforce the integrity principle and delegate the confidentiality one to the underlayer mechanisms of the OS (e.g. processes sandboxing) or the network (e.g. TLS/SSL) layer. *Cevt* is composed of two sections, the *<payload/>* and *<integrity/>*. The payload encloses the workflow characteristics such as a *messageID* (message identifier), *orchesID* (orchestrator running the workflow) or an *activityID*. The latter is especially important as there could be as many as parallel instances of the same activity running in the workflow, because a mobile workflow as we previously state can be distributed among different mobile devices. Finally, the *sa* value covers the information the SA will use to perform the security controls and actions. In this example the control is enforced remotely and the security risk is HIGH.

Regular activities, as previously said, only encapsulate invocations to local or remote resources (capabilities or devices, among other possibilities). However, the SA is a virtual activity and, inside, it contains five different modules which processes the received events and executes the required actions: the discriminating module, the integrity module, the evaluation module, the decision module and the universal subscriber. The first module is focused on managing the SA lifecycle. The next three modules verify the integrity of the received *Cevt*, and perform the security controls and actions. The last one is charge of tracking the application of the security actions.

The first module, the discriminating module, manages the three states of the SA: *secure, ready and reset.*

The Secure state is employed to ensure that the SA shuts down as secure as it started (it is the first and last state in the SA lifecycle). It implies the application of a restricted profile where all existing security tools from the middleware are activated. This state maintains the workflow execution as secure as possible if the proposed security framework is deactivated or detected (due to on-device or network exploitations).

The transition to a Ready state occurs when all the *At* in the workflows to be executed have been instantiated, which means that they are ready to invoke resources in the middleware and to trigger *iNotify()* primitives. In Ready state the SA receives the *Cevt* triggered from all existing *At* which are part of the same workflow instance. Only if the Integrity module (we are presenting below) guarantees the received *Cevt* are not tampered or compromised (using the *icheck* parameter, see next paragraphs) the SA maintains in this state. If *Cevt* are proved to be tampered, the SA operation is considered to be compromised; it changes to secure state and gets shutted down. When an Activity finishes its execution (At') and initiates a *eNotify(Cevt)* message (due to the injection of a security control in this activity), the reception of this event leads the SA to change its state to a Reset State which finally triggers a *setsatype()* primitive and/or an OS API invocation (once security controls and actions have been evaluated). This primitive

Fig. 6 Structure of Control Event

```
<?xml version="1.0" encoding="UTF-8" ?>
<controlevent>
   <payload>
     <workflow>
       <messageid>8901</messageid>
       <orchesid>alicelocalph</orchesid>
       <taskinstanceid>12345<taskinstanceid>
       <actinstanceid>12345</actinstanceid>
     </workflow>
     <sadata>
       <satype>REMOTE</satype>
       <classev>HIGH</classev>
     </sadata>
   </payload>
   <integrity>
     <hashf>SHA256</hashf>
     <secretvalue>0000</secretvalue>
     <mac>d2f344c728ea9b2d9fdca6c0644ccaabaefca06b5c997e4bdb45cd55227ac8a6</mac>
   </integrity>
</controlevent>
```

informs the orchestrator (or the original *At*) whether it has to apply some modification in the context of the selected security actions. In order to clarify this explanation, Fig. 7 shows the finite state machine.

Taking as reference the scenario described in Sect. 3.1, the local orchestrator of the workflow will receive a *setsatype(<remote>)* invocation, indicating the ecommerce transaction must continue in other (remote) mobile device (the new location should be negotiated by the SA with the corresponding remote SA). Thus, the orchestrator invokes the middleware with the adequate parameters and establishes a connection with an external device where Bob could use the remote authentication and authorization service and finishes the transaction. If in the process of establishing this new connection, a non-authorized change in execution flow occurs, the SA (which, as we are seeing, tracks the application of the proposed security actions) changes to a secure state and blocks the execution.

At the same time that the discriminating module manages the SA lifecycle, the other modules in the SA are in charge of evaluating the security controls, selecting the most adequate security actions and tracking their application. In particular, the Integrity Module of the SA guarantees that messages, *Cevt*, are not modified in a non-authorized way and enforces a BiBa-based integrity model (i.e. a secure broadcast protocol based on signatures) for the entire system (as only transactions employing this protocol are considered). To perform the integrity exam, this module reads the <integrity/> value of the *Cevt* by processing a message authentication code (MAC). The MAC allows the SA to verify the authenticity of the received messages, comparing messages from activities which are part of the same workflow. In fact, as activities could also be executed in different mobile devices (as execution may be distributed) it is important for the overall model to provide anti-tampering capabilities, as disabling or transferring in a non-authorized way some activities or operations could produce the entire workflow compromise. The *<hashf/>* value define the one-way function and secret

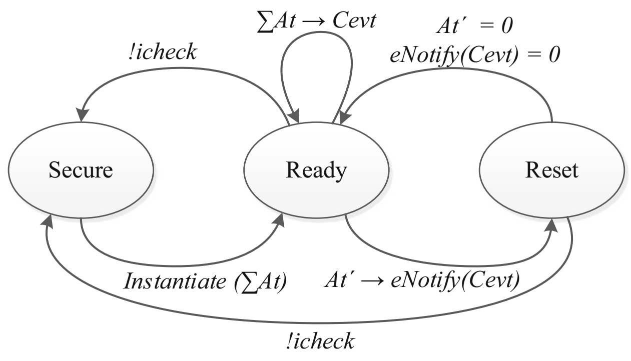


Fig. 7 Finite state machine for SA

value the *secretkey* used to create the MAC, which is shown below. The generation and delivery of secret keys in each workflow is out of the scope of this article, so we assume that mechanisms exist to carry out them when the workflow is initiated. After a successful MAC check an internal boolean variable, named *icheck*, is generated and set to 1, otherwise is set to 0. This variable is employed to discard (or not) the received *Cevt*, as well as to put the SA in secure state.

The behavior of the next two modules is designed to perform the security controls (i.e. evaluate the relevant security indicators previously defined) and, based on the obtained results, the information provided in the *Cevt* and the offered capabilities by the OS, select the most appropriate security actions to be taken. The Evaluation Module is in charge of the first task. The Decision Module performs the second process. The definition of the relevant security indicators, as well as the definition of the implemented solutions to support the decision-making process are not the objective of the paper, so they are not discussed nor analyzed.

Finally, the Universal subscriber (US) keeps track of the selected security actions that were already moved to the orchestrators using the *setsatype()* primitive. In order to do that, the SA invokes the *subsextsa()* primitive using the Low-level OS API. Briefly, the US monitors if the transmitted orders to the external orchestrator were successfully or unsuccessfully carried out or if there were exceptions triggered by the activities or the middleware (e.g. not valid secret keys or SSL Bumpings). In this case, US detects the problem and moves the system to a secure state (the security framework is shutted down). An administrator should verify the systems at this point and apply the most adequate action.

4.2 Detailed explanation of transactions

Depending on the security situation, the actions to be taken and the evolution of the context and/or the workflow execution, the transaction between the execution system and the security framework may be short or very heavy and time consuming. The next figures present some uses cases in order to illustrate the detailed operation of the proposed solution.

Figure 8 shows an example where the system is initialized and security actions are transferred in a successful way.

As can be seen, just after the workflow initiates its execution the SA reads all security controls existing locally and remotely via the Low-level OS (1). Afterwards, a certain Activity with an injected security control generates a *eNotify(Cvet)* message directed to the SA (2). The latter processes the event, evaluates the security indicators,

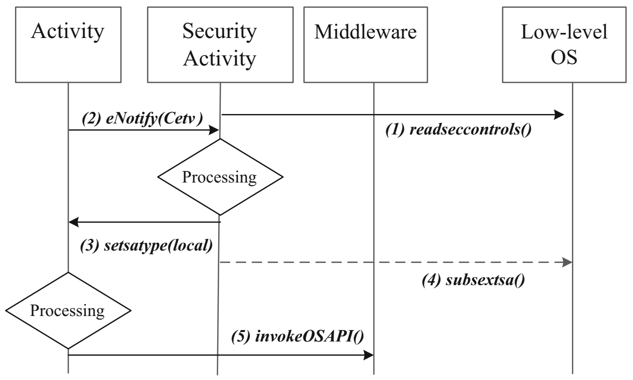


Fig. 8 Successful transaction

selects the security actions to be taken and, in this case, sends back a *setsatype(local)* message (3) which informs the activity (orchestrator) about the actions to be performed. After this process, the SA subscribes to notifications from the Low-level OS in order to track the application of the selected actions (4). The orchestrator (activity) modifies the execution along the received orders and invokes directly the Middleware (5) in order to continue with the execution, with no further action as security actions are successfully applied.

In a new use case (Fig. 9), a certain Activity sends to its SA a Control event (1). After processing it, the SA decides it is necessary to employ remote resources in the context of the selected security actions. This situation, in fact, is similar to scenario described in Sect. 3.1. In the next step (2) The SA(A) informs an external Security Activity(B) which controls the remote resources to be available for serving request from the original activity. Thus, SA(B) accepts and configures (3) in the middleware the necessary ports and authentication info received from original activity (<workflow/> and <integrity/> values), in order to validate and initiate the connections in the data layer. In step (4), the Security Activity A subscribes to future exceptions created by the Middleware related this specific workflow instance (if a fail occurs the entire system will be moved to secure state). In step (5) the SA(A) receives acknowledge of a successful creation of the required configurations in system B (e.g. open ports, certificates and bindings) needed to support the transactions of the original activity. In step (6), the SA(A) informs the orchestrator about the security actions to be taken, indicating the authorized usage of remote resources, including the required parameters (e.g. TLS configuration, TCP/IP data and WSDL). Finally, the orchestrator invokes the OSAPI (7) in data plane in order to move transaction to the remote mobile device.

In the last use case (see Fig. 10), we are assuming there was a compromise situation where communications were affected by a Man-in-the-Middle (MiTM) attack which disrupts TLS communication, so the Low-level OS notifies

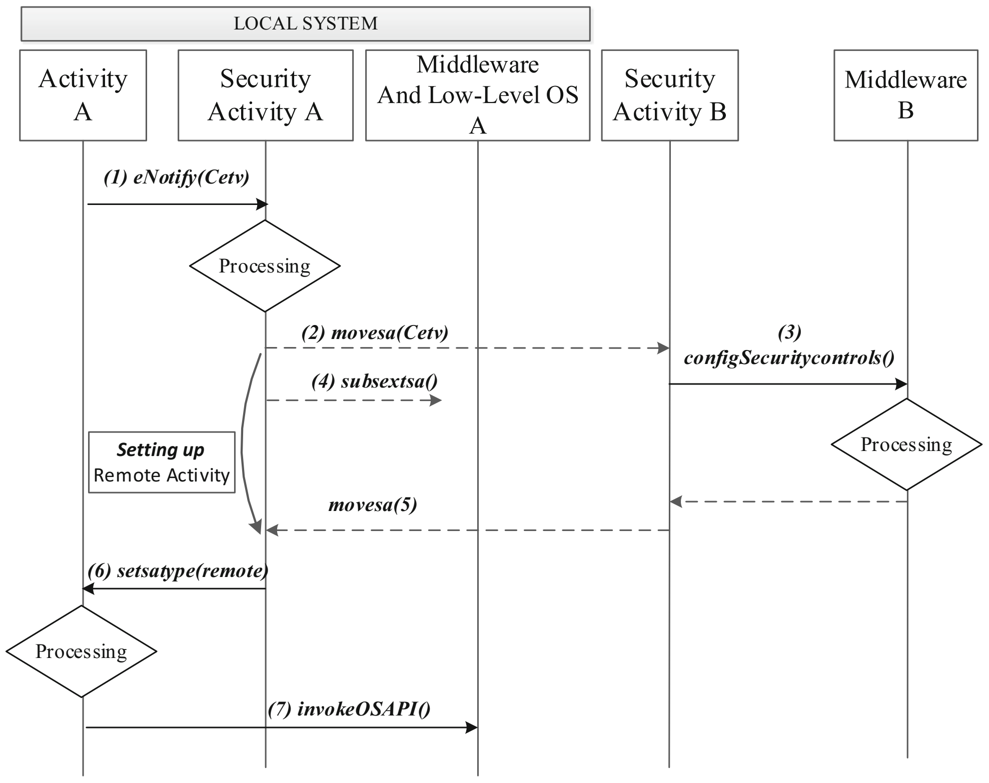


Fig. 9 Successful transaction using remote resources

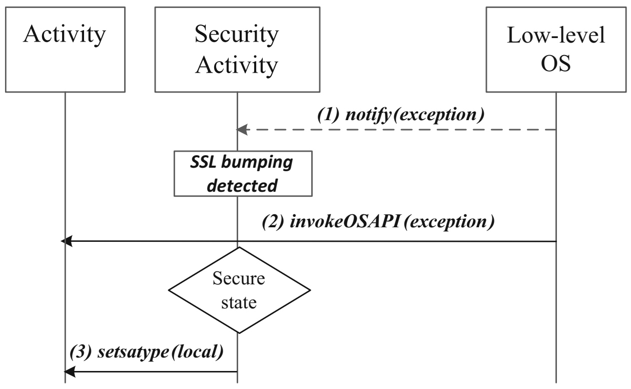


Fig. 10 Failed transaction

the exception (1) to the SA and it detects the corresponding attack. In that moment, the exception also reaches original activity (2), as it was the instance that invokes the middleware, and the system is moved to secure state by the SA, which sends a signal (3) to the original activity to use regular security tools.

5 Security framework implementation and validation

The security framework whose development has been described in the previous section has been implemented in a prototype in the form of a mobile application for Android operating system. Although, a deep and complete validation of the proposed solution would require the provision of some experiments over iOS as well, some technical problems prevent the realization of these analyses. Briefly, since iOS11, the capabilities of private APIs were drastically decreased; and measures of workflow execution delays are not accurate as iOS does not allow tightly controlling the actions that apps sent to back ground execute within the TCP/IP stack.

This section describes the libraries that have been used and the set of tests that have been done with this prototype in order to evaluate the correct injection of flexible security controls in the workflow execution.

5.1 Implementation details

For the detection of events that can cause changes in workflow behavior, the *BroadcastReceiver* Android resource is used to register for an event and to trigger when the event happens. The receiver gets triggered once the event happens or when a custom broadcast is sent. This allows the security framework to receive the events produced by the workflow tasks that are running.

For transferring some workflow fragments or components for remote execution, we need to serialize Android classes and transfer them. To perform the required processes of *marshalling* and *unmarshalling*, we use the Android's *Parcelable* interface, which provides more performance and reduce the useless code comparing to *Serializable* interface.

Finally, access to secure communications are done through HTTPS by using *Voley* library, in cases of RPC-type operations, such as fetching structure data, images, JSON, etc. For large streaming operations such as a voice calls we use the *DownloadManager* systems service instead.

To evaluate the performance of the prototype in the correct injection of flexible security controls we created three services in which to apply security actions in real time according to the preventive, corrective and adaptive scenarios presented in this work.

The first service developed (Serv#1) performs periodic balance inquiries in *Bitcoin*, using the *bitcoinj* library. It also enables periodic transfers between two accounts. Method *wallet.getBalance()* checks account balance and method *wallet.sendCoins(…)* enables transferring *Bitcoins*. This service is intended to be a service that runs without interruption, thus, if an event stating that the mobile terminal is running out of battery (*android.intent.action.BATTERY_LOW*) is received, the security framework will perform a *marshalling* of the java classes used to make the regular transfers and will allow its execution in another Android terminal. To facilitate the execution of this service multiple times, as the low-level battery event cannot be triggered artificially, we use the Mock low-battery broadcast intent.

The second service (Serv#2) is a location service based on *Geofences*. A geographic area is defined and saved into the application by employing the *Google Geofencing API for Android*, and *exit Geofence* events are monitored, so that users leaving the perimeter activate the injection of corrective security mechanisms. These security mechanisms imposed by the security framework are translated into a transfer of the running activities to an external server, which requests permission to the Android application to access the location information, through the *Google Location Services API*, part of *Google Play* services. In this way, a user with a compromised mobile phone can be located, even if the local executing workflow is suspended, due to device hijacking.

The third implemented service (Serv#3) allows establishing a call, and during the duration of this call, checks the security of the Wi-Fi network. Once the Wi-Fi network is detected insecure, the security framework asks another participant workflow to make a Wi-Fi *Hotspot* with Internet connection through the 4G network, so that the application can connect to this protected Wi-Fi network seamlessly, to continue with the call. The detection of changes in the access network is done through the library *ReactiveWiFi*, for Android, by observing changes in the WPA Supplicant state. We also use the *ConnectityManager* Android class to fail over the data network when connectivity to Wi-Fi network is lost.

5.2 Experiments and results

A series of experiments are carried out over these implemented services, so findings are below. First experiment was designed to evaluate the effectiveness of the proposed security framework. In that way, during this experiment the success rate of the designed security method was measured. Second experiment was employed to compare the proposed solution to existing methods in the state of the art in terms of resource consumption. Finally, the third experiment evaluates the execution delay of the proposed solutions.

As said, the objective of the experiment Experiment#1 is to evaluate the effectiveness of our security framework, measuring the number of correctly executed services when an injected security control triggers some security actions. We define as correct execution that the service can continue to offer its functionality once the security mechanisms have been established, that is, a cryptocurrency operation can be performed (Serv#1), the individual can be located outside his *Geofence* (Serv#2) and the call can be restored through the new secure access point (Serv#3).

We execute 50 times each of the services and force the security framework to intervene in its execution. In Fig. 11 we counted in percentage values the number of correct executions for each of the services, and the aggregated value of executions in the framework.

As can be seen in Fig. 11, Serv#2 and Serv#3 have lower successful ratios. In Serv#2 this is due to connectivity problem in device mobility (4% of cases) and authentication problems to receive device location (4% of cases). In Serv#3 this is due to faults in the library recognizing the unsafe state of the initial Wi-Fi network (6% of cases) and problems in the execution of the functionality provided by the *ConnectityManager* class to programmatically perform the connection to the new Wi-Fi hotspot (4% of cases).

The objective of Experiment#2 is to compare the performance of the proposed solution to the existing methods in the state of the art. In order to do that, we evaluated the system load by periodically checking the evaluation points present in the workflows. On the other hand, we implemented the same services but following a monolithic software design. The performance of both implementations in term of resource consumption was compared. Besides, in order to provide a more exhaustive analysis, a third service implementation was considered: a design through dynamic loading of components (based on OSGi technology -Open Services Gateway initiative-) [23].

For this we analyze the Android processes. As the process that runs the security mechanisms is implemented as an Android Service, we can measure the CPU and memory consumption of this process separately to the application implementing each Service#. We run Serv#1, #2 and #3 interacting with the security framework, through the processes of Service binding and Service unbinding of Android. The following figure measures the average consumption of CPU (User + kernel modes) and Memory (Allocated Memory) measured by the Android Monitor of Android Studio 2.3.3 for each Serv#, considering the case in which they communicate with the security framework through the interfaces provided by the Android Service (***Fw***), the case in which the security actions have not been provided by the framework but are programmed in each Serv#, as a monolithic application (***Mon***); and the case in

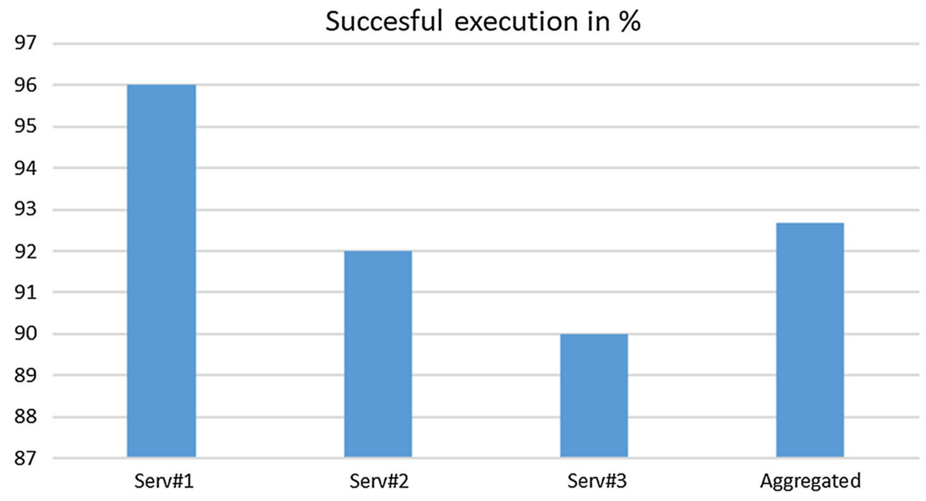


Fig. 11 Measurements of successful service execution and aggregated percentage

which the security actions are provided by components loaded in a dynamic way (***Dyn***). The system used for the measurements has been a Nexus 5X API 23 (Android 6.0).

As it can be seen in the Fig. 12, the percentage of CPU used is greater in the case including the security framework active than in monolithic designs; but is lower than in system of component dynamic load. However, the memory consumption is lower than in any other case (by not having loaded the security functions until it is not necessary to make a connection with the Framework, and by only requiring lightweight components to operate). Requirements of Serv#3 are greater than those of Serv#2, and this in turn, is higher than those of Serv#1. This is because the computational complexity of service 3 is greater, having to secure the multimedia traffic in real time. In Serv#2, although it is also multimedia traffic since the user's position on a map is represented, the hardware and software requirements are smaller.

The objective of Experiment#3 is to determine the impact on the injection of security control, through measures of delay in execution. For this purpose, some measurements are performed in the different evaluation points present in the services, considering the starting point of the measurements the *binding* process with the interface provided by the security framework and the end point of the measurement, the beginning of the invocation of the security control. In this way we collect in the measurement only the invocation impact of the security framework from the workflows in execution and the decision process of the control action to be implemented. The results of the execution delay can be seen in Table 1.

It can be observed how the delays imposed by the communication with the security framework are quite similar, and less than 400 ms, which implies that the decision on the preventive, corrective or adaptive actions necessary to establish a better control of the workflow security are taken almost immediately, without diminishing the Quality of Experience of the local agent.

Table 1 Results of the third experiment

Service	Execution delay (ms)
Serv#1	285
Serv#2	329
Serv#3	387

6 Conclusions

This paper addressed the problem of modeling security controls in mobile and distributed workflow executions, and how to enhance mobile workflow execution in the security scope. We proposed a framework to inject security controls in workflows, by modeling security as control points present in executing mobile workflows. The results obtained in the injected security controls are employed to make decisions (corrective, preventive and adaptive actions) in a flexible way, depending on the execution system capabilities and the workflow context, in order to enhance the mobile execution.

As an extension of a previous paper [4], we evolved our previously defined workflow model considering control and security events, and defining the transaction between the execution system and the security framework, including some mechanisms to support the usage of remote resources in the context of the selected security actions.

The security framework was implemented in a prototype in the form of a mobile application for Android. To

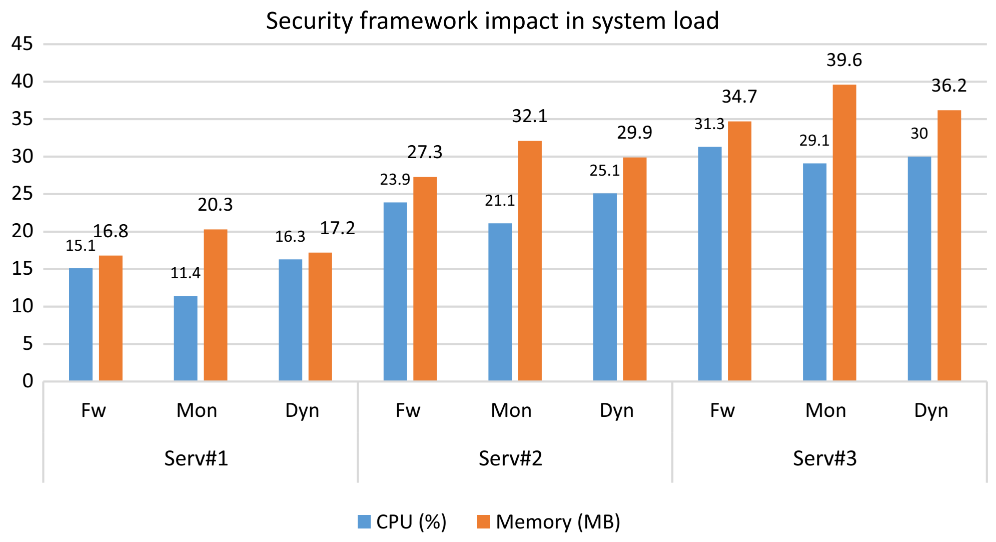


Fig. 12 Measurements of security framework impact in system load

evaluate the prototype's performance, we created three services for the application of security actions in real-time according to the preventive, corrective and adaptive scenarios.

The effectiveness of our security framework has been evaluated my measuring the number of correctly executed services when an injected security control triggers some security actions. We found that in more than 90% of the cases the security framework correctly intervened in service execution and the execution was successful, even in the case of the most complex services.

The evaluation of the system load was performed by periodically checking the evaluation points present in the workflows. We compared our framework solution to other state of the art solutions: monolithic software design and dynamic loading of components (based on OSGi technology). Results in this experiment shown that, although the percentage of CPU is greater in the case including the security framework, the memory consumption is lower than in the case of a monolithic approach, due to the optimization in loading the security functions just before they are needed; and also lower than in the dynamic loading approach, as the framework only requires lightweight components to operate.

Finally, the delay in execution was measured to determine the impact on the injection of security control. It was determined that a delay of 400 ms implies that the decision on the preventive, corrective or adaptive actions necessary to establish a better control of the workflow security are taken almost immediately and does not diminish the Quality of Experience of the local agent.

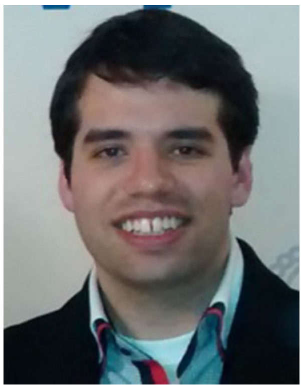
As conclusions, the proposed framework facilitates the injection of security features in mobile workflows, and provides good flexibility in the integration of transversal security aspects, decoupled to mobile workflow executions, without diminishing the Quality of Experience of the local agents.

Acknowledgements The research leading to these results has received funding from the Ministry of Economy and Competitiveness through SEMOLA project (TEC2015-68284-R) and from the Autonomous Region of Madrid through MOSI-AGIL-CM project (Grant P2013/ICE-3019, co-funded by EU Structural Funds FSE and FEDER). Borja Bordel has received funding from the Ministry of Education through the FPU program (Grant Number FPU15/03977).

References

1. Sánchez, B. B., Alcarria, R., de Rivera, D. S., & Sánchez-Picot, Á. (2016). Enhancing process control in industry 4.0 scenarios using cyber-physical systems. *JoWUA, 7*(4), 41–64.
2. Bordel, B., Alcarria, R., Robles, T., & Martín, D. (2017). Cyber–physical systems: Extending pervasive sensing from control theory to the Internet of Things. *Pervasive and Mobile Computing, 40,* 156–184.
3. La Polla, M., Martinelli, F., & Sgandurra, D. (2013). A survey on security for mobile devices. *IEEE Communications Surveys & Tutorials, 15*(1), 446–471.
4. Alcarria, R., Robles, T., Morales, A., & Cedeño, E. (2014). Resolving coordination challenges in distributed mobile service executions. *International Journal of Web and Grid Services, 10*(2–3), 168–191.
5. Bordel, B., Sánchez de Rivera, D., Sánchez-Picot, Á., & Robles, T. (2016). Physical processes control in industry 4.0-based systems: A focus on cyber-physical systems. In C. García, P. Caballero-Gil, M. Burmester, & A. Quesada-Arencibia (Eds.), *Ubiquitous computing and ambient intelligence. UCAm I 2016, IWAAL 2016, AmIHEALTH 2016. Lecture notes in computer science* (Vol. 10070). Cham: Springer.
6. Bordel, B., Alcarria, R., Sánchez-de-Rivera, D., & Robles, T. (2017). Protecting industry 4.0 systems against the malicious effects of cyber-physical attacks. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 161–171). Cham: Springer.
7. Parker, F., Ophoff, J., Van Belle, J. P., & Karia, R. (2015). Security awareness and adoption of security controls by smartphone users. In *2015 Second international conference on information security and cyber forensics (InfoSec)* (pp. 99–104). Cape Town.
8. Wen, Z., Cala, J., & Watson, P. (2014). A scalable method for partitioning workflows with security requirements over federated clouds. In *2014 IEEE 6th international conference on cloud computing technology and science*, Singapore, 2014 (pp. 122–129).
9. Wen, Z., & Watson, P. (2013). Dynamic exception handling for partitioned workflow on federated clouds. In *2013 IEEE 5th international conference on cloud computing technology and science (CloudCom)* (vol. 1, pp. 198–205).
10. Chang, V., Kuo, Y. H., & Ramachandran, M. (2016). Cloud computing adoption framework: A security framework for business clouds. *Future Generation Computer Systems, 57,* 24–41.
11. Marcon, D. S., et al. (2013). Workflow specification and scheduling with security constraints in hybrid clouds. In *2nd IEEE latin american conference on cloud computing and communications, Maceio* (pp. 29–34).
12. Chen, H., Zhu, H., Qiu, D., Liu, L., & Du, Z. (2015). Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. *IEEE Transactions on Parallel and Distributed Systems, 28*(9), 2674–2688.
13. Hussain, S, Sinnott, R. O., & Poet, R. (2016). A security-oriented workflow framework for collaborative environments. In *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin (pp. 707–714).
14. Hussain, S., Sinnott, R. O., & Poet, R. (2016). Security-enabled enactment of decentralized workflows. In *2016 Proceedings of the 9th International Conference on Security of Information and Networks (SIN '16)* (pp. 49–56). New York, NY, USA: ACM.
15. Peng, T., Chi, C. -H., Chiasera, A., Armellin, G., Ronchetti, M., Matteotti, C., Parra, C., Kashytsa, A. O., & Varalta, A. (2014). Business process assignment and execution in mobile environments. In *2014 International conference on collaboration technologies and systems (CTS)*, Minneapolis, MN (pp. 267–274).
16. Younis, Y. A., Kifayat, K., & Merabti, M. (2014). An access control model for cloud computing. *Journal of Information Security and Applications, 19*(1), 45–60. https://doi.org/10.1016/j.jisa.2014.04.003. (**ISSN 2214-2126**).
17. Gao, B., He, L., Lu, X., Chang, C., Li, K., & Li, K. (2015). Developing energy-aware task allocation schemes in cloud-assisted mobile workflows. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure*

computing; pervasive intelligence and computing, Liverpool, 2015 (pp. 1266–1273).
18. Deng, S., Huang, L., Taheri, J., & Zomaya, A. Y. (2015). Computation offloading for service workflow in mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems, 26*(12), 3317–3329. https://doi.org/10.1109/TPDS.2014.2381640.
19. Abrishami, S., Naghibzadeh, M., & Epema, D. H. J. (2012). Cost-driven scheduling of grid workflows using partial critical paths. *IEEE Transactions on Parallel and Distributed Systems, 23*(8), 1400–1414.
20. Zeng, L. F., Veeravalli, B., & Li, X. R. (2015). SABA: A security-aware and budget-aware workflow scheduling strategy in clouds. *Journal of Parallel and Distributed Computing, 75*, 141–151.
21. Li, Z., Ge, J., Yang, H., Huang, L., Hu, H., Hu, H., et al. (2016). A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. *Future Generation Computer Systems, 65*, 140–152. https://doi.org/10.1016/j.future.2015.12.014. **(ISSN 0167-739X)**.
22. Tang, X., Li, K., Zeng, Z., & Veeravalli, B. (2011). A novel security-driven scheduling algorithm for precedence constrained tasks in heterogeneous distributed systems. *IEEE Transactions on Computers, 60*(7), 1017–1029.
23. Alcarria, R., Robles, T., Morales, A., & Gonzalez-Miranda, S. (2012). Flexible service composition based on bundle communication in OSGi. *KSII Transactions on Internet and Information Systems, 6*(1), 116–130. https://doi.org/10.3837/tiis.2012.01.007.



Borja Bordel Ph.D. Candidate by ETSIT de Technical University of Madrid. Master in Science, Technical University of Madrid. Researcher in GISAI Group UPM, along with Spanish and European Projects. Interests: cyber physical systems, mobile communications, IOT.

Ramon Alcarria Assistant Professor at the School of Geomatic Engineering, Technical University of Madrid. Ph.D. in Telecommunication Engineering from the Technical University of Madrid, Department of Telematic Engineering. Researcher in the Technical University of Madrid (since 2007), collaborating in several research projects. Master degree in Networks and Services Engineering (Department of Telematic Engineering, DIT). Interests: Service Composition, service oriented architectures internet of things, web of things future internet, FI-WARE location-based services, social mobility.

Augusto Morales Mobile Security Architect at Check Point Software Technologies Ltd. Ph.D. Telematic Systems Engineering from Technical University of Madrid. M.Sc. Networks and Services Engineering, Technical University of Madrid. Research interest: mobile security, network security, IOT, mobile services. IEEE senior member.

Ignacio Castillo has been working for 20 years in Computing and telecommunications industries [Datacenter Dynamics, RedUno-TELMEX, IFE, DICINET], as well as in universities as tenure & invited professor. He has participated in more than 50 national and international projects as a team member, or leader on technical and management positions. He participates with the Mexican NB for ISO/IEC 27000 & ISO/IEC 85500 standards. He wrote 3 books, 22 journal & conference papers, 14 technical reports and +100 articles for magazines and newspapers. Castillo has lectured +70 keynote and invited talks. He is referee for IEEE LA Transactions, IEEE conferences and IEEE Smart Cities initiative. Castillo is an IEEE Senior Member, IEEE Computer Society DVP and an IEEE COMSOC LA Region board member. He was a member of the Board of Governors of IEEE Computer Society.