A passive testing approach for security checking and its practical usage for web services monitoring

Ana Rosa Cavalli, Azzedine Benameur, Wissam Mallouli, Keqin Li

A Passive Testing Approach for Security Checking and its Practical Usage for Web Services Monitoring*

Ana Rosa Cavalli
Institut Telecom / TELECOM & Management SudParis
9, rue Charles Fourier
F-91011 Evry Cedex, France
ana.cavalli@it-sudparis.eu

Azzedine Benameur
SAP Labs
805 Avenue du Docteur Maurice Donat, F-6000
Mougins Cedex, France
azzedine.benameur@sap.com

Wissam Mallouli
Montimage EURL
39 rue Bobillot
F-75013 Paris Cedex, France
wissam.mallouli@montimage.com

Keqin Li
SAP Labs
805 Avenue du Docteur Maurice Donat, F-6000
Mougins Cedex, France
keqin.li@sap.com

ABSTRACT

To achieve a meaningful business goal, Web services are combined and connected together based on a predefined workflow. In this distributed configuration, tasks are executed by different entities usually managed by different business partners which makes the security monitoring of the whole business process complex. Indeed, the application of classical monitoring methods is not suitable in this kind of service oriented architecture (SOA) where execution traces collection is generally distributed and security requirements implicate several Web services. In this paper we propose a passive testing approach for SOA, encompassing a non-intrusive module that gathers selected traces for web services in both cases of centralized and decentralized workflows, and also a passive tester that analyzes the distributed collected traces and deduces a verdict concerning the respect of the Web services to their security requirements. Finally we apply the proposed methodology to a Loan Origination Process using BPEL workflow.

Keywords

Web Services, Passive Testing, Security Checking, Nomad Formal Language.

1. INTRODUCTION

Web service technology is defined by the W3C [14] as a software system designed to support interoperable machine-to-machine interaction over a network. It provides the possibility to integrate business applications and connect business processes across company boundaries. A business process can then be composed of individual Web services that belong to different companies. In other words, a business process is a network of Web services without any global supervision system. To leverage Service Oriented Architecture (SOA) [12], services are connected together to achieve more complex tasks using a workflow specification language [2]. Currently, Business Process Execution Language for Web Service (BPEL4WS, denoted BPEL in the following) [6] is the de facto standard to describe the interactions of the individual Web service in both abstract and executable ways.

In SOA, communications are made through XML-Based messages called SOAP [13] messages. These communications are distributed and their security requirements implicates several Web services. Thus, ensuring this security in such distributed configuration is challenging. Testing is a mean to provide guarantee that the security properties are holding when the system is implemented.

Formal testing allows to check the respect of the security requirements of a system. It can be either active or passive. Active testing permits to validate a Web service implementation by applying a set of test cases and analyzing its reaction. It implies that we have a global control on the service architecture which is difficult to perform in a dynamic topology in the case of decentralized workflows for instance. Besides, the active testing becomes difficult to perform when the network is built from components (services) that are running in their real environment and cannot be interrupted or disturbed. In this situation, there is a particular interest in passive testing techniques in which traffic flows of deployed Web services are monitored. This testing consists in analyzing collected data according to some functional and security requirements described in a formal specification.

In this paper, we rely on passive testing techniques to study web services in both cases of centralized and decentralized workflows in order to detect any security flaw. Our main contributions are the following: (i) A trace collection mechanism for both centralized and distributed web services. (ii) Formalization of security properties/policies using Nomad formalism. (iii) A detailed implementation of the proposed mechanism (trace collection and passive testing tool). (iv) A running example with security analysis, and selected properties formalized.

The remainder of this paper is organized as follows. In section 2, we discuss the related work tackling with Web services workflows monitoring. Section 3 presents the distributed collection mechanism of the web services traffic in both cases of centralized and decentralized architectures. Section 4 presents the methodology and tool to analyze this global trace by comparing it to the security requirements described in Nomad formal language. In section 5

*The research leading to these results has received funding from the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement no 215995.

we apply our methodology on an industrial case study: Loan Origination Process. Section 6 concludes the paper and presents future work.

2. RELATED WORK

Most of works related to Web services reasoning focus on two related but distinct problems. The first one is the formal description of Web services composition. One of the proposed approaches is to use planning techniques on behavioral models as in [7, 9, 10]. For instance, [9] proposes a methodology to specify Web services composition into Petri nets and provides decision procedures for Web service simulation, verification and composition. Other works, such as [10], rely on transition rule systems that model the interactions among services at the knowledge level. This allows us to avoid the explosion of the search space due to the usually large and possibly infinite ranges of data values that are exchanged among services. The authors of [7] proposes to build the behavioral Web services model by automatically translating existing process descriptions, such as BPEL ones, into timed automata system. This system specification is used later for model-based testing purpose.

The second problem is the property verification on Web services in order to guarantee that deployed applications satisfy a set of requirements and temporal properties (for instance, the absence of deadlocks). It is usually argued, for instance by [11], that existing automated model-checking tools can support these tasks under the condition that components' behavior and their interactions are described by formal models. Web services development reinforces the need of tools to improve their reliability and their security respect. Several works [5, 15] tried to deal with the functional behavior of Web services. [5] is devoted to monitoring component-based software systems whose behavior is modeled using a formalism based on Petri nets. In addition, [15] relies on existing BPEL specifications and examine how to translate them into a transition rule formalism. In this paper, we propose a formal approach to develop a monitoring tool dedicated for Web services workflow security checking. We claim that our methodology is innovative and do not generate any fault positives.

3. PRELIMINARIES

3.1 Background on BPEL Workflow for Web Services

A workflow could be defined as a collection of tasks organized to accomplish some business process [4]. A task can be performed by one or more software systems (e.g., web services), one or a team of humans, or a combination of these. In addition to a collection of tasks, a workflow defines the order of task invocation or condition(s) under which tasks must be invoked, task synchronization, and information flow (dataflow). Workflow management systems (WFMSs) are used to provide the ability to specify, execute, report on, and dynamically control workflows.

Traditionally, the workflow management and scheduling is carried out by a single centralized workflow management system. This WFMS is responsible for enabling task execution, monitoring workflow status, and guaranteeing task dependencies. However, in an electronic commerce environment with inter-organizational workflows, since the systems are inherently distributed, heterogeneous, and autonomous in nature, decentralized workflow management systems are used.

Therefore with respect to our work, the collection mechanism of execution traces has to work with both the centralized and decentralized architectures.

3.1.1 Centralized Workflow

A typical example of centralized workflow is a business process defined by BPEL4WS [2]. BPEL Business Processes offer the possibility to aggregate web services and define the business logic between each of these service interactions. It is also said that BPEL orchestrates such web service interactions. Each service interaction can be regarded as a communication with a business partner. Such an interaction is potentially two sided: the process invokes the partner and the partner invokes the process.

In BPEL, there exist a couple of simple activities with the purpose of consuming messages from and providing messages to web service partners. These activities are the receive activity, the reply activity and the invoke activity. All these activities allow exchanging messages with external partners (services). At the same time, BPEL provides means to structure the business logic according to business needs, using sequence activity, if-else activity, repetitive activities, etc.

From the trace collection point of view, each service (business partner) can have a local view of execution information (e.g., SOAP message exchange) related to the business process, and the orchestrator, i.e., the BPEL engine, has a global view.

3.1.2 Decentralized Workflow

From a system point of view, the collaborating enterprises constitute an autonomous decentralized system since they pursue a common goal as a whole but each of them is highly independent. The features of this kind of collaboration are as follows:

- On the one hand, it is required that business processes of collaborating partners should be integrated seamlessly in order to ensure high operation efficiency;
- On the other hand, it is required that only necessary information should be exchanged across organizational boundaries. The internal structures of the business process should remain as black-box because the technical know-how and commercial secrets of a company are embedded in its business processes.

Therefore, integrated business process management of this loosely coupled and dynamically changed partnership does not require a centralized business process model that will be shared and maintained by all collaborating partners. Instead, it is required that business processes of an enterprise are designed, implemented and managed by the enterprise itself. The function as well as the monitoring and control of a business process is wrapped as externally accessible service, which will be integrated with the business processes of the collaborating enterprises on demand as well as on the fly. In summary, business processes are modeled and executed in a decentralized manner.

Correspondingly, several decentralized workflow management systems are proposed, such as [1] and [8].

From the trace collection point of view, in such an architecture, no single workflow management agent has the global view of the workflow. The distributed pieces of execution information of the workflow need to be correlated to obtain a global picture.

3.2 Security Rules Specification for BPEL

To formally specify security properties that the BPEL-based workflow has to respect, we rely on an interpretation of Nomad language [3]. Nomad stands for 'Non Atomic Actions and Deadlines' model. It allows to express privileges on non atomic actions or tasks performed within a workflow. It combines deontic and tempo-

ral logics and can describe conditional[1] privileges and obligations with deadlines. The main features of Nomad model are to provide means to specify:

- Privileges (permission, prohibition or obligation) associated with non atomic actions/tasks.
- Conditional privileges, which are privileges only triggered when specific conditions are satisfied.
- Privileges that must be fulfilled before some specific deadlines.

Nomad formal model is designed to regulate the nature and the context of tasks that can be performed within different systems or organizations workflow. Intended to be generic, Nomad do not formally define any atomic action it can rely on, so that it can be applied to case studies with different characteristics. In this paper, we instantiate Nomad to fulfill passive testing aspects and we define then an atomic action as the occurrence of an event within the studied system workflow. This event (emission or reception of a message between two system components) can be detected by the trace collection unit and is stored in a log file. Some constraints on message parameters values are also considered in the action syntax description:

$Event$ (*Par*$_1$ *op Val*$_1$, *Par*$_2$ *op Val*$_2$, ... , *Par*$_n$ *op Val*$_n$)

Where:

- $Event$ represents a message exchanged between two system entities.
- Par_i (i $\in$ {1, ... , n} are the message parameters. These parameters represent the relevant fields in the message (n is the number of these relevant message fields). Some of these parameters are always considered, such as the message type, the sender identifier (IP address) and the destination identifier (IP address).
- Val_i (i $\in$ {1, ... , n}) are the possible parameters values.
- *op* is an operator that allows to compare Par$_i$ and Val$_i$. op $\in$ {=,$\neq$, <, >, $\leq$, $\geq$, $\ni$}.

DEFINITION 1. *(Abstention of doing an action)*
If A is an action that can be performed within a system S, then $not(A)$ (which means "the non occurrence of A") is an action.

DEFINITION 2. *(Non-atomic action)*
If A and B are actions that can be performed within a system S, then $(A; *; B)$ (which means "A followed by B"), $(A; B)$ (which means "A followed immediately by B") and $(A\&B)$ (which means "A in parallel with B") are non-atomic actions.

DEFINITION 3. *(Formula)*
If A is an action then $start(A)$ (A is being started), $doing(A)$ (A is being performed) and $done(A)$ (A has been finished) are formula.

DEFINITION 4. *(Formula)*

- *If α and β are formula then $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ and $(\alpha \leftrightarrow \beta)$ are formula.*
- *If α is a formulae then $\oplus^n\alpha$ (in the n next messages/events in the trace, α will be true) and $\ominus^n\alpha$ (in the n previous messages/events in the trace, α was true) are formula.*
  For instance, if we are in the event number 'nb', $\oplus^n\alpha$ means that α will be true for a certain event i such as ($nb < i < nb+n+1$). While, $\ominus^n\alpha$ means that α was true for a certain event i such as ($nb - n - 1 < i < nb$).
- *If α is a formulae then $O^d\alpha$ (α was true d units of time ago if $d \leq 0$, α will be true after d units of time if $d \geq 0$) is a formulae.*
- *If α is a formulae then $O^{<d}\alpha$ (within d units of time ago, α was eventually true if $d \leq 0$, α is eventually true within a delay of d units of time if $d \geq 0$) is a formulae.*
- *If α is a formulae then $\bigodot^{<d}\alpha$ is formula. $\bigodot^{<d}\alpha$ is to be read α will be true (if $d \geq 0$) or was true (if $d \leq 0$) during d units of time.*
- *If α and γ are a formula, then $(\alpha|\gamma)$ is a formulae: in the context γ, the formulae α is true.*

DEFINITION 5. *(More details)*

- *If $n = 1$, then $\oplus^1\alpha$ can be denoted $\oplus\alpha$ and means: immediately in the next event in the trace, α will be true.*
- *If $n = 1$, then $\ominus^1\alpha$ can be denoted $\ominus\alpha$ and means: immediately in the previous event in the trace, α was true.*
- *If $n = \infty$, then $\oplus^\infty\alpha$ means: at certain point in the future in the trace, α will be true.*
- *If $n = \infty$, then $\ominus^\infty\alpha$ means: at certain point in the past in the trace, α was true.*

Notice that using Nomad formalism, we deal with a discrete time. The choice of the unit of time can be very important and depends on the studied system.

DEFINITION 6. *(Temporal modalities)*
If α is a formulae then $\Box\alpha$ (α is necessary) and $\Diamond\alpha$ (α is possible) are formula.

DEFINITION 7. *(Deontic modalities)*
If α is a formulae then modality $\mathcal{P}(\alpha)$ (α is permitted), $\mathcal{F}(\alpha)$ (α is forbidden) and $\mathcal{O}(\alpha)$ (α is mandatory) are formula.

DEFINITION 8. *(A security rule)*
We define a security rule as a formulae with the following format: $\mathcal{R}$ $(\alpha|\beta)$ where $\mathcal{R} \in \{\mathcal{P},\mathcal{F},\mathcal{O}\}$ and α and β are formula. $\mathcal{P}$ $(\alpha|\beta)$ (resp. $\mathcal{F}$ $(\alpha|\beta)$, $\mathcal{O}$ $(\alpha|\beta)$) means that it is permitted (resp. prohibited, mandatory) to have α true when context β holds.

4. TRACE COLLECTION MECHANISM FOR SOA

Web services can be seen as black-box, where only the interfaces and input/output parameters are known. The communication is made through SOAP messages where these interfaces are called and parameters are sent.

In general, to collect execution traces on a running system, we need to install observation points (called also probes) into specific strategic points. These observations points aim to collect data exchanged between relevant entities. The collected traces are usually stored in one or many trace files depending on if we are dealing

[1]Most privileges do not apply unconditionally. Thus, we need to model privileges that are only active in specific contexts.

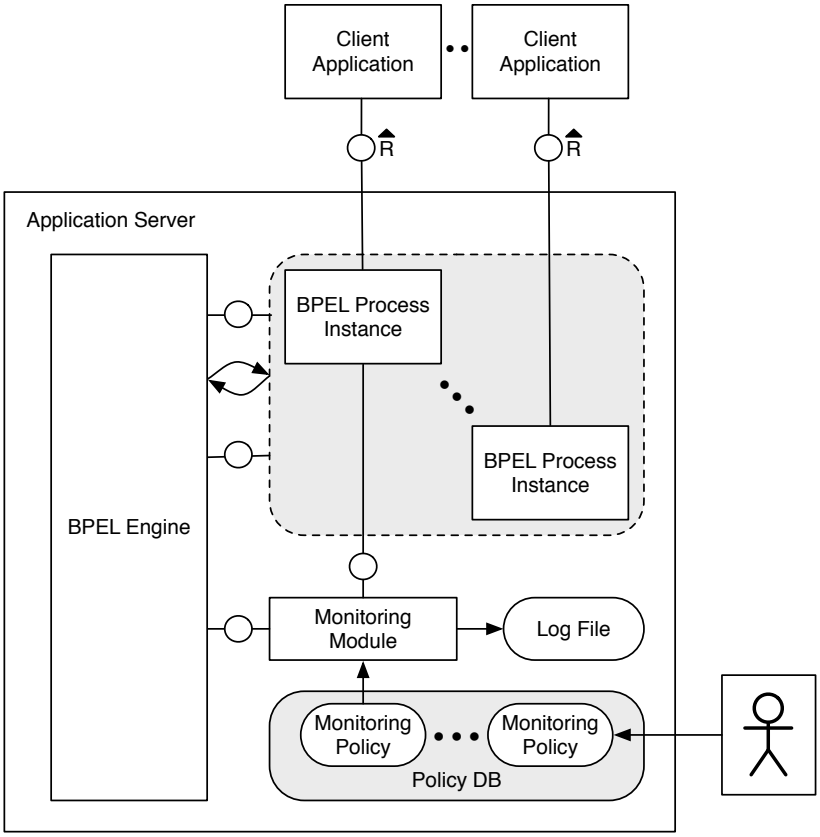

Figure 1: Trace Collection Module

with centralized or distributed workflows. In the case of Web services monitoring, we rely on integrated modules within each BPEL engine that collects local traces. We retrieve then trace files that describe the communication between distributed Web services in a system.

The architecture of our trace collection module is depicted in Figure 1. The module is placed at each workflow engine layer. It logs selected elements of a SOAP message, which are defined in the logging policy file. In the case of centralized workflow, all the message exchanges are handled by the BPEL workflow engine and only a trace collection module is needed.

Table 1 gives an example on how the logging policy is expressed. Each service is identified by its namespace, the name of the XML element follows. The first audit rule depicted is to select the content of the node $< username >$ from a web service identified by the namespace $http://ws1.com$ and to select the same node from another web service identified by the namespace $http://ws2.com$ which is distinct from the first one.

The second audit rule specifies to log the content of the node $< LoanRequest >$ from a web service identified by the namespace $http://ws1.com$ and to select all the nodes (using $*$) from the web service identified by the namespace $http://ws3.com$. For readability we use two rules but they can be expressed in one line using the spacing separator. The policy can also be expressed using de facto standards such as Xpath.

5. PASSIVE TESTING METHODOLOGY

5.1 Preliminaries

Our passive testing methodology for security checking is divided into three main steps:

- The definition of passive testing architecture: This architecture depends on the nature of the workflow. Collection modules are placed at each workflow engine layer. The traces files are then merged (based on each event timestamp) into one file that describes all the exchanged messages within the whole system. Notice that in the case of a centralized workflow, only one file is generated from the beginning.

- The description of the system security requirements using a formal specification language: the description concerns the security rules that the studied system has to respect. We rely in this paper on Nomad language introduced in section 3.2. This can be done by an expert of the Web service under test that understands in details its security requirements.

Audit rule 1	$logNodList = http://ws1.com \mid username \quad http://ws2.com \mid username$
Audit rule 2	$logNodList = http://ws1.com \mid LoanRequest \quad http://ws3.com \mid *$

Table 1: Example of a Logging policy

- The security analysis: based on the security policy specification, the passive tester has to perform security analysis on the global trace file to deduce a global verdict. This verdict is *PASS* if the system trace respects the specified security policy and *FAIL* if it does not. The *INCONCLUSIVE* verdict is possible if the tester can not extract the necessary information from the collected traces in the case of a short trace for example. If the trace is long enough (or if the traffic collection is continuous), we can claim that it describes the global behavior of the system and consequently the verdict concerns the system conformance according to its security policy.

5.2 Passive Testing Tool

To automate the process of scanning the captured traces in order to detect possible violations of security rules, we rely on a passive testing tool developed by Montimage. A high level description of this tool is given in Figure 2.

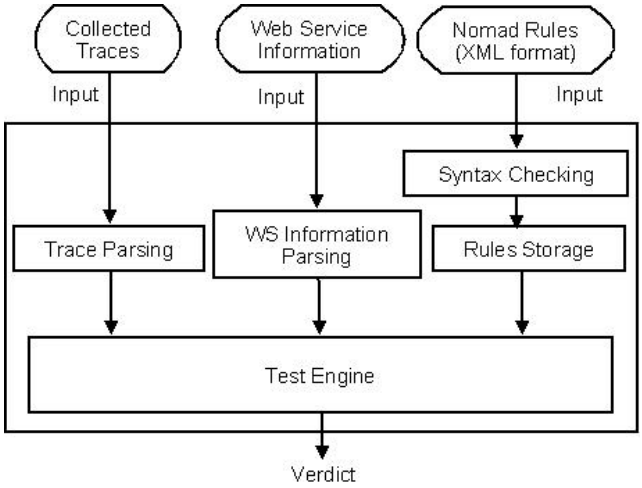

Figure 2: Passive Testing Tool

This tool allows automated analysis of the captured traces to determine if the given Nomad security rules are satisfied or not. It takes as input three different files:

- The system security rules defined/described in Nomad formal language and written in XML format.
- The traces captured by the collection module.
- And information on the Web Service that is being observed. This information represents data of interest (mainly event fields' names) that are relevant to the automated analysis of the captured traces. It indicates the fields of interest (defined also in the logging policy file) so that only these are extracted and stored by the tool.

In order to use the tool, first the security rules and the data of interest need to be defined. This is performed by an expert of the Web service. Then, the security rules are written in XML format (see Figure 3) to make them easier to be interpreted by humans and software.

The context part of the rule is denoted by an <if> tag that identifies the triggering event. An event is a set of conditions that need to be satisfied for a given message exchange. When analyzing a security rule, we have the triggering event, called 'reference' event, and the other events, called 'current' events, that should occur before or after the triggering event (depending on the rule). The events that need to be verified on the 'current' events are found in the <then> tag. Conditions are of the form:

(<variable>)(<operation>)(<variable> | <value>)

meaning that a <condition> compares a <variable> to <value> or another <variable> through an <operation>. Variables can refer to the 'reference' event or the 'current' event. This allows comparing the two. Values can be strings or numbers and operations can be: contains, is equal to, does not contain, is different than, is less than, is less than or equal to, etc.

The second step is to parse the captured trace to analyze it according to these security rules. The verdict obtained for a security rule can be either PASS, FAIL or INCONCLUSIVE meaning respectively that all events where satisfied, that at least one event was not satisfied or that it is not possible to give a verdict due to the lack of information in the trace.

```
...

<rule name="NOMAD RULE N">
<if>
  <event verdict="FALSE">
    <condition>
      <variable>EventType</variable>
      <operation>=</operation>
      <value>calculateLoan</value>
    </condition>

    ...
  </event>
</if>
<then type="BEFORE" max_skip="-1" max_time="10">
  <event verdict="TRUE">
    <condition>
      <variable>EventType</variable>
      <operation>=</operation>
      <value>storeLoan</value>
    </condition>

    ...
  </event>
</then>
</rule>

...
```

Figure 3: XML Format for Defining a Nomad Security Rule

6. CASE STUDY: LOAN ORIGINATION PROCESS

6.1 Introduction

The Loan Origination Process (LOP) describes a customer wanting to buy a bundled product. Several external and internal ratings need to be obtained by the processing clerk in order to check the credit worthiness of the customer (internal rating, Credit Bureau). Once the credit worthiness of the customer has been positively established, the bank selects a bundled product and submits it to the customer. If the customer is satisfied by the proposed product, both parties come to an agreement and sign a contract.

Figure 4 illustrates the different steps of the scenario, with the actors and the roles involved. After the customer makes a loan request:

- (1) His identity is checked by the pre-processing clerk. Then, two parallel rating engines start and the customer worthiness is establish by two separate entities.

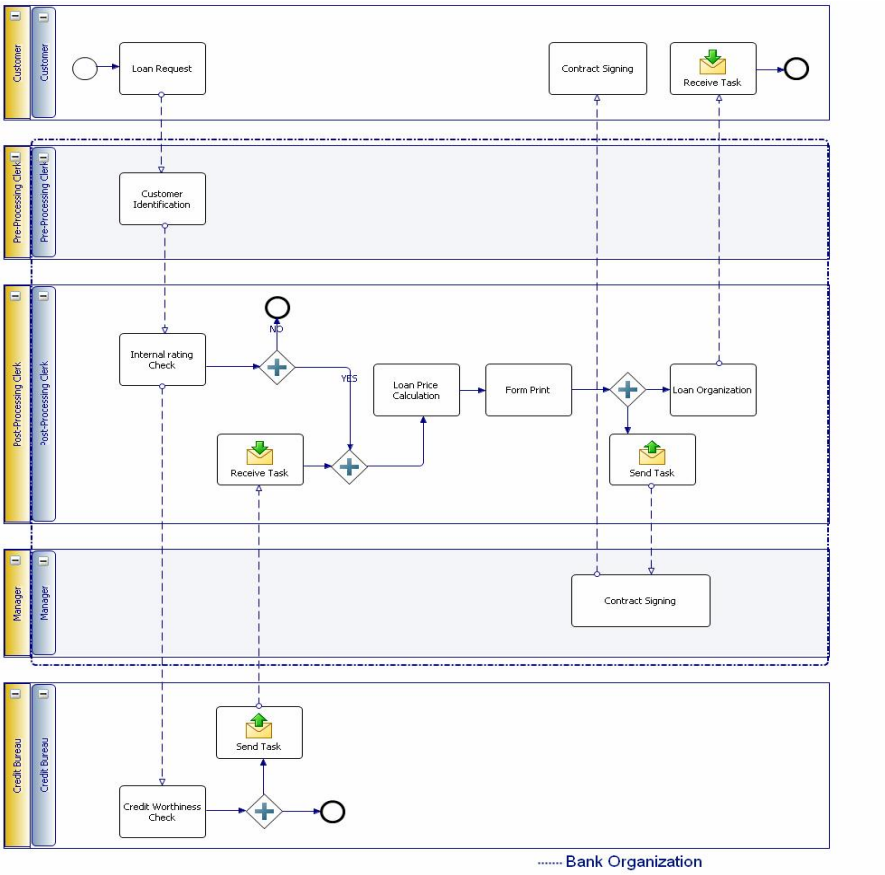

Figure 4: Loan Origination Process Workflow

- (2) The credit worthiness of the customer is checked by querying a third party, namely Credit Bureau.
- (3) The internal rating checks the customer worthiness with regards to his history in the bank. In case of negative result, the manager can intervene.
- (4) The bank calculates the price of the loan using the bank Internal Computer System
- (5) The bank and the customer come to an agreement.

This case study in a banking context is very interesting from the security and privacy point of view. It involves different organizations dislocated geographically and organizationally which implies that some properties can be satisfied locally and no necessary hold on the global workflow.

6.2 Architecture and Implementation

The implementation of the scenario is based on the Service Oriented Architecture paradigm [12]. We assign several services to the different entities: front office, back office, back office of the manager. So far, we have implemented a simplified version of a workflow having excluded a manager approval task.

- The front office is in charge of identifying the customer, storing the loan request and finalizing the contract. These functions are performed respectively by the following Web services: 'WS Authentication', 'WS Store Loan' and 'WS Loan Calculation'.
- The risk assessment is done by a parallel invocation of the bank internal rating engine and a third party represented by WS Internal Rating and WS External Rating.
- The back office is modeled as an interface of the WS Calculation Offer.

The logical architecture of the LOP implementation is depicted in figure 5. Four layers are needed to fully decouple the business logic from the functional implementation. A database layer makes a clear separation between the database of the bank computing the internal rating, and the credit bureau database providing an external assessment. The Rules layer enables decoupling of the business rules from the code in such a way that if regulations (or practices) of the partner evolve, the only layer to update is the rules layer. The Web service layer is complemented by the orchestration layer that contains the BPEL specification of the process that is interpreted by a WFMS. The latter describes how web services works together and on which interface.

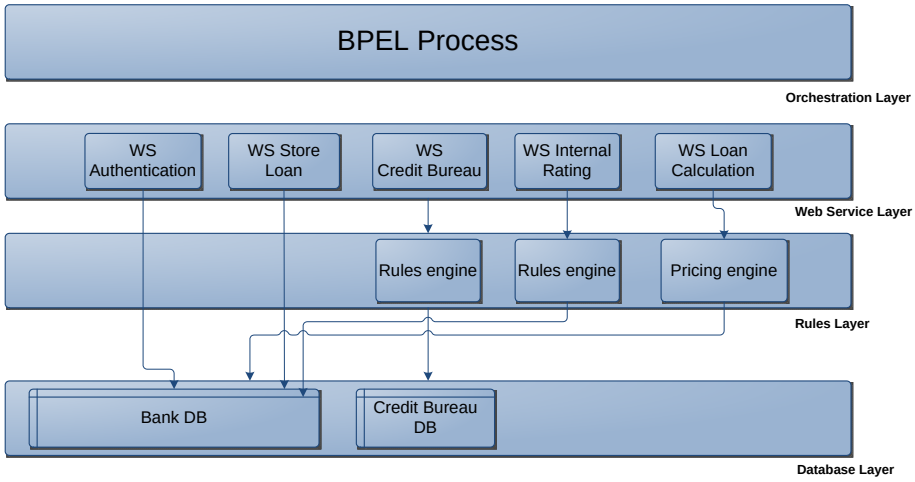

Figure 5: Layered Implementation Achitecture of The Loan Origination Process Workflow

6.3 Security Analysis

We want to check several security properties that the LOP process has to respect. The first step consists in formalizing them using Nomad formal language:

1. (Security Rule 1) Separation of duty: between the pre and the post processing clerks. The loan processing is divided into two main phases: pre-processing and post-processing. To avoid frauds the pre-processing clerk should not be the same employee as the post-processing clerk.

$$\mathcal{F}(start(Event(EventType = calculateLoan, Processor = user))| \ominus^{\infty} done(Event(EventType = storeLoan, Processor = user)))$$

where user = "http-8080-Processor"+"i" and i $\in$ {1,...,30}

2. (Security Rule 2) Authentication: to ensure that the user is authenticated before executing the workflow. The employee should be authenticated before performing any operation. In the LOP, this is obtained by contacting a *Security Token Service* which issues a security token to the service client. The service client contacts the web service with the provided token and the service can decide whether the client is a trusted entity or not.

$$\mathcal{F}(start(Event(EventType = Any, Processor = user))| \neg \ominus^{\infty} done(Event(EventType = WSAuthentication, Processor = user)))$$

where user = "http-8080-Processor"+"i" (i $\in$ {1,...,30}) and Any $\in$ {WSLoanStorage, WSInternalRating, WSCreditBureau, WSLoanCalculation, WSWFTerminator }

3. (Security Rule 3) Deny of service attack avoidance: if a same clerk/cus-tomer has more than 3 loan requests within the same minute, we consider him/her as a potential malicious clerk/customer and we might want to investigate later.

$$\mathcal{F}(start(Event(EventType = WSLoanRequest, Processor = user))| O^{<-1min} done(Event(EventType = WSLoanRequest, Processor = user)); *; Event(EventType = WSLoanRequest, Processor = user)); *; Event(EventType = WSLoanRequest, Processor = user)))$$

where user = "http-8080-Processor"+"i" and i $\in$ {1,...,30}

4. (Security Rule 4) Timeout property: calculateLoan should be called within 10 minutes after storeLoan is called.

$$\mathcal{P}(start(Event(EventType = calculateLoan, Processor = user_1))|O^{<-10min} done(Event(EventType = storeLoan, Processor = user_2)))$$

where $user_1$ = "http-8080-Processor"+"i" (i $\in$ {1,...,30}), $user_2$ = "http-8080-Processor"+"j" (j $\in$ {1,...,30}) and i $\neq$ j.

6.4 Trace Collection

In order to be able to check the properties expressed in 6.3. We first identify the web services involved, and then the elements contained in the SOAP message that will be required to use passive testing. The trace collection policy depicted in table 2 is based on the following reasoning.

- Separation of Duty: the pre-processing clerk identifies him self on WS Authentication, and then the release of the loan is done by the post-processing clerk on WS Store Loan. The XML schema for these messages contains a tag element $< employeeName >$. Therefore the audit rules for this properties will log the element $< EmployeeName >$ from the two namespaces respectively belonging to WS Authentication and WS Store Loan. (see audit rules 1 and 2)

- Authentication: Each workflow instance has a unique ID. We need to check that for any given workflow ID there is a respective successfully message exchange with WS Authentication. The audit rule 3 expresses that we have to log messages from all the services (namespace=http://*) that contain the LoanID element. Then, we use audit rule 4 that allows to verify that the authentication is ok.

- Deny of service attack avoidance: each loan request has a unique number and is associated to a pre and post processing clerk. We here use the complete execution trace over multiple workflow instances and the elements $< LoanID >$, $< EmployeeName >$. (see audit rules 5 and 6)

In figure 6 presents some trace lines which show the brokered authentication mechanism.

Audit rule 1	$logNodList = http://authentication.ws.ebusiness.prototype.serenity.crnce.sap.com \mid EmployeeName$
Audit rule 2	$http://loanstorage.ws.ebusiness.prototype.serenity.crnce.sap.com \mid EmployeeName$
Audit rule 3	$logNodList = http://* \mid LoanID$
Audit rule 4	$logNodList = http://authentication.ws.ebusiness.prototype.serenity.crnce.sap.com \mid ClientSSN$
Audit rule 5	$logNodList = http://authentication.ws.ebusiness.prototype.serenity.crnce.sap.com \mid EmployeeName$
Audit rule 6	$logNodList = http://authentication.ws.ebusiness.prototype.serenity.crnce.sap.com \mid LoanID$

Table 2: Trace Collection Policy

```
2008-10-28 16:25:23,812 [http-8080-Processor18]
INFO  com.sap.crnce.serenity.patterns.security.
a2.logging.SecureLogHandler  - SecureLoggingMod
ule: http request from Address: 127.0.0.1 to Ad
dress: http://localhost:8080/axis2/services/Sec
urityTokenService

2008-10-28 16:25:23,812 [http-8080-Processor18]
INFO  com.sap.crnce.serenity.patterns.security.
a2.logging.SecureLogHandler  - SecureLoggingMod
ule: action performed -> http://schemas.xmlsoap
.org/ws/2005/02/trust/RST/Issue

(...)

2008-10-28 16:25:27,046 [http-8080-Processor16]
INFO  com.sap.crnce.serenity.patterns.security.
a2.logging.SecureLogHandler  - SecureLoggingMod
ule: http request from Address: 127.0.0.1 to Ad
dress: http://localhost:8080/axis2/services/WSA
uthentication

2008-10-28 16:25:27,046 [http-8080-Processor16]
INFO  com.sap.crnce.serenity.patterns.security.
a2.logging.SecureLogHandler  - SecureLoggingMod
ule: action performed -> urn:AuthenticateClient

(...)
```

Figure 6: Example of Captured Traces for LOP Process

```
<rule name="NOMAD RULE 1">
<if>
  <event reference="TRUE" verdict="FALSE">
    <condition>
      <variable>EventType</variable>
      <operation>=</operation>
      <value>StoreLoan</value>
    </condition>
  </event>
</if>
<then type="AFTER" max_skip="100" max_time="infinity">
  <event verdict="TRUE">
    <condition>
      <variable>EventType</variable>
      <operation>=</operation>
      <value>CalculateLoan</value>
    </condition>
    <condition>
      <variable>Processor</variable>
      <operation>different</operation>
      <variable type="REFERENCE">Processor</variable>
    </condition>
  </event>
</then>
</rule>
```

Figure 7: XML Format for The First Nomad Security Rule

6.5 Passive Testing Results

The security rules described in Section 6.3 were evaluated in the traces provided by SAP obtained from the point of observation depicted in the Figure 1. The security rules are defined using the XML format supported by the passive testing tool (see figure 8 for the first Nomad security rule), defining the types of events to inspect and evaluate in the trace. This file, in conjunction with the trace file, where provided as input for the passive testing tool.

The verdicts obtained were PASSED for all the security rules, except for three cases where the security rule number 2 was applied (2 INCONCLUSIVE verdict and one FAIL verdict) and for four cases where the security rule number 4 was applied taking into account a 10 minutes timeout value (FAIL verdict). A more detailed review of the tool output showed that the FAILED results of the fourth rule actually correspond to *false positive* verdicts and correct events in the trace were detected as errors by the experiment. Indeed, a mistake was embedded in the XML design of the security rule where the timeout was assigned to 100 seconds and not 10 minutes.

The results obtained with the second security rule shows the difficulty, in some cases, of applying passive testing techniques to an application such as the LOP process. Indeed, at the beginning of the captured trace, some operations are performed and the authentication of users is not logged (but perhaps this authentication is done before the beginning of the trace capture). This indicates that the definition of the security rule is not sufficiently precise to be relevant in the case of too short traces that do not provide all the needed data information.

Notice also that the security rule number three was never applied in the collected trace since no malicious was acting during the trace capture.

In general the results of the experimentation showed that the techniques used worked correctly on the captured traces and the approach is suitable for applying on Web services such as the LOP process.

7. CONCLUSIONS AND FUTURE WORK

Passive testing has shown to be well adapted to test systems, or components, which are in operation in their real environment and cannot be interrupted or disturbed. In this paper, we presented a passive testing approach and its application for security checking of web services based on the SOA architecture. In particular, the passive testing is applied to a case study, the LOAN Origination Process using BPEL workflow. The security properties have been

described using the Nomad language, well adapted to describe permission, prohibition and obligations and includes time constraints. The security properties to be evaluated have been provided by SAP and their evaluation has been carried out using a passive testing tool developed by Montimage. The implementation of the Loan Origination Process has also been provided by SAP.

Several experiments have been performed on the service implementation. The application of these techniques showed that passive testing techniques can be applied to industrial cases studies to help improve the reliability of the proposed services and that the techniques and tools are scalable.

As future work, we plan to explore and evaluate new types of security rules in this framework. The XML representation of the Nomad security rules need also to be refined to be able to express and test more complex security rules. Regarding the passive testing tool, we plan to adapt it to be able to analyze SOAP packets online. This will also allow the tool to be used to detect that the protocol exchanges occur as expected during the operation of the Web service.

8. ADDITIONAL AUTHORS

No additional authors.

9. REFERENCES

[1] V. Atluri, S. A. Chun, and P. Mazzoleni. A chinese wall security model for decentralized workflow systems. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 48–57, New York, NY, USA, 2001. ACM.

[2] Business process execution language v2.0, 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.

[3] F. Cuppens, N. Cuppens-Boulahia, and T. Sans. Nomad: A Security Model with Non Atomic Actions and Deadlines. In *CSFW*, pages 186–196, 2005.

[4] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure, 1995.

[5] I. Grosclaude. Model-based monitoring of software components. In R. L. de Mántaras and L. Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI)*, pages 1025–1026. IOS Press, 2004.

[6] M. B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition*. Packt Publishing, 2006.

[7] M. Lallali, F. Zaidi, A. Cavalli, and I. Hwang. Automatic timed test case generation for web services composition. *Web Services, European Conference on*, 0:53–62, 2008.

[8] F. Montagut, R. Molva, and S. T. Golega. The pervasive workflow: a decentralized workflow system supporting long running transactions. *IEEE Transactions on Systems, Man and Cybernetics Part C - Applications ans reviews Volume 38 N3: Special issue on Enterprise services computing - May 2008*, 2008.

[9] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM.

[10] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated composition of web services by planning at the knowledge level. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI*, pages 1252–1259. Professional Book Center, 2005.

[11] G. Salaün, L. Bordeaux, and M. Schaerf. Describing and reasoning on web services using process algebra. In *ICWS*, pages 43–. IEEE Computer Society, 2004.

[12] Oasis standard: Reference model for service oriented architecture v1.0, 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html.

[13] Simple object access protocol 1.1, 2000. http://www.w3.org/TR/soap/.

[14] World Wide Web Consortium. http://www.w3.org/.

[15] Y. Yan, M.-O. Cordier, Y. Pencole, and A. Grastien. Monitoringweb service networks in a model-based approach. *Web Services, European Conference on*, 0:192–203, 2005.