I. J. Computer Network and Information Security, 2019, 10, 26-36
Published Online October 2019 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2019.10.04

Security Policy Modelling in the Mobile Agent System

Hassan RAZOUKI
LMACS Laboratory, Faculty of Science and Technology, University Sultan Moulay Slimane, Beni Mellal, Morocco
E-mail: razouki.hassan@gmail.com

Received: 02 July 2019; Accepted: 25 August 2019; Published: 08 October 2019

***Abstract*—**The mobile agent security problem limits the use of mobile agent technology and hinders its extensibility and application because the constantly progressed complexity and extension at the level of systems and applications level increase the difficulty to implement a common security system as well as an anticipated security policy.

Ontology is considered one of the most important solutions to the problem of heterogeneity. In this context, our work consists of constructing mobile agent domain security ontology (MASO) in order to eliminate semantic differences between security policies in this domain. We use the OWL language under the protected software to construct this ontology. Then, we chose the WS-Policy standard to model security policies, these policies are structured in forms of security requirements and capabilities. To determine the level of semantic correspondence between security policies we are developing an algorithm called "Matching-algorithm" with Java language and two APIs (Jena API and Jdom API) to manipulate the MASO ontology and security policies.

***Index Terms*—**Mobile agents, security, security policy, ontology, semantics.

I. Introduction

The power of mobile agent technology in solving complex problems results from the fact that agents, thanks to their autonomy, mobility and adaptability, can achieve their goals in a flexible way by using local and/or remote interaction with other agents on the network. However, the flexibility and mobility of the mobile agent poses a serious security problem that has hindered its expansion. [1]. The implementation of a security policy may require, on the one hand, the protection of the resources and data of the host machines, and on the other hand, the preservation of the integrity and confidentiality of the agents themselves and their communications [2].

In this context, the interest in the protection and security of mobile agents and the services offered by platforms has increased within organizations. As a result, different security policies have been developed, and different security standards have been proposed. This has led to heterogeneity in the exploitation of these security policies by different entities.

Mobile and service agents are autonomous entities, potentially heterogeneous, of diverse origins, and free to enter and leave the system whenever they wish [3]. In such a scenario, interoperability problems frequently occur that require specific resolution techniques. Our effort is focused on solving these kinds of problems by focusing on the heterogeneity of security policies between these entities.

In order to achieve interoperability and resolve issues of heterogeneity between the security policies of the mobile agent and the platforms visited, semantic integration is necessary. Security ontologies, at present, are considered as the next trend to solve heterogeneity problems, as it offers a shared knowledge which is able to prevent communication and interaction failure among mobile agents, this failure is due to their heterogeneous security properties [4]. And this is the reason which pushed us to produce a common security domain ontology that will present concepts, relations, integrity constraints and rules on which agents and platforms could collaborate.

The ontology we have proposed has a twofold objective: first, the establishment of formal knowledge on security in mobile agent-based systems, and second, the use of ontology facilitates automatic analysis of the semantic compatibility between the agent's security policies and the platforms visited. We have chosen to model security policies using a W3C standard called WS-Policy. We add semantic annotations using this ontology to describe security requirements and capabilities. Indeed, we have structured the security policy in two parts:

- Security requirements: Allows you to specify the different security settings necessary for the secure execution of a mobile agent
- Security capacity: represents a set of specifications, protocols, algorithms..., to satisfy a security requirement.

To determine whether a platform is capable of securely executing an agent, on the one hand, the functional aspects of the platform should satisfy the functional needs of the agent to perform their task, on the other hand, the agent's security requirements must be satisfied by the platform's security capabilities, so the platform's security

requirements must also be satisfied by the security capabilities used by the mobile agent.

The objective of this article is to build an ontology in the field of mobile agent security in order to eliminate the semantic differences that exist between security policies in this field. We use the OWL language under Protected 4 [5] to build this ontology. Then, we chose the WS-Policy standard [6] to model security policies in terms of security requirements and capabilities. To determine the level of semantic correspondence between security policies we are developing an algorithm called "Matching-algorithm" with Java language and two APIs (Jena API and Jdom API) to manipulate MASO ontology and security policies.

The rest paper is discussed as follows. In section II, previous work on security ontologies is discussed. Section III, how to model security policies in mobile agent systems with the WS-Policy standard and the security ontology (MASO) we have constructed. Section IV presents the process of matching security policies based on the MASO ontology, we also demonstrate the importance of this solution using an example of the interaction between a mobile agent and an execution platform (how to apply the semantic matching algorithm). Finally, Section V concludes the paper.

II. Related Work

Ontology is considered one of the most important solutions to solve the problem of heterogeneity. In the literature several safety ontologies have been developed, targeted safety ontologies are classified and grouped into three main categories: generalized safety ontologies, specific safety ontologies, and diverse safety ontologies [4].

1. The generalized security ontologies aimed to cover security features, which had formed explicit domain terminology for dissimilar stakeholders. This category of ontology pays attention to security development and contribution to the knowledge database with general logical perceptive without human intervention [4]. Some of the generalized security ontologies were cloud computing security taxonomies [7], ontology-based Security [8] and ontology-based multi-agent model based on information security system [9].
2. The specialized security ontologies focused on a range of computational models having variables from general terminologies related to security requirements application-based security, network, risk and web services, etc. These ontologies were alienated into five subcategories with respect to special aspects of security [4]. Web Services (WS) and Web Ontology Language (OWL) based Security Ontologies [10,11]. Network Security Ontologies [12,13]. Risk-based Security Ontologies [14]. Application-based Security Ontologies [15,16].
3. Miscellaneous Security Ontologies [4]. There are numerous ontologies which cannot be sited in any of the aforementioned categories; thus such types of ontologies are placed in the miscellaneous category. Some of the specialized security ontologies were (Information Security Measuring Ontology (ISMO) [17], Vulnerability-Centric Modeling Ontology [18], Cyber Ontology [19], Security Toolbox: Attacks and Countermeasures (STAC) Ontology [13], Ontological approach toward cybersecurity in Cloud Computing [20], Cloud Ontology [21], Security Ontology Driven Multi-Agent System Architecture: Cloud Data Storage [22].

Subsequent related researches show the importance of using security ontologies in several domains (cloud computing, web service, networks, application...). In the field of security of mobile agent systems, Hacini's approach [15] is considered one of the most important solutions to solve the heterogeneity problem. This solution uses an ontology to eliminate semantic differences in security policy objects, attributes, and data structures to facilitate mobile agent interoperability. The limitations of this approach lie in the fact that ontology is used only in a communication scenario between mobile agents and platforms. Indeed, this ontology does not solve the problem of specifying security policies in the mobile agent system, nor the problem of heterogeneity between the security policies of the mobile agent and the platforms visited. Finally, this ontology does not provide a solution to describe the specific security needs of each agent, nor the security capabilities provided by the platforms.

In the following, we will show how to specify semantic security policies for mobile agents and platforms that offer services to agents, as well as how to semantically match these two security policies.

III. Security Policy in Mobile Agent Systems

The discovery and selection of the most appropriate platforms for the secure execution of mobile agents are important steps in our approach. We consider platform discovery to be the location of published platforms that satisfy certain functional properties of the agent to perform their task. The selection of platforms corresponds to the evaluation and ranking of platforms already discovered in order to identify those that best meet the security requirements of the mobile agent. Indeed, each platform must have a functional description of the services offered to mobile agents, as well as a non-functional description concerning the security of each service offered by the platform [23].

In order to be able to use the security policy in the selection process of the platforms visited by the mobile agent, they must be modeled and attached to the services when they are published and to the mobile agents when they are created. We adapted the WS-Policy specification [6] to express security requirements and capabilities and proposed a security policy model specific to the mobile

agent system.

A. Semantic need in policy correspondence

The major problem with the use of WS-Policy is that the correspondence between policies is based solely on a syntactic comparison; the intersection of policies can reject potential partners in many cases, even with compatible policies. We demonstrate the usefulness of semantic comparison through the following example:

A mobile agent requires data confidentiality and provides authentication capability:

- Security requirement: a mobile agent which requires a constraint on the confidentiality of the data produced by the visited platform and requires the encryption of this data with the 3DES algorithm
- Security capability: the mobile agent offers an authentication mechanism with an X.509 digital certificate.

A platform requires the authentication of visiting mobile agents and offers symmetric encryption capability:

- Security requirement: the visiting mobile agent must be authenticated
- Security capacity: the platform has an XML-Encryption encryption specification.

In the above scenario, if we use the WS-Policy standard to represent and match the agent's security policies and platform. The comparator makes a syntactic comparison between character strings to determine whether the platform capacity can satisfy the agent's requirement, and the agent's capacity can satisfy the platform requirement. The comparator necessarily concludes that these two policies are not compatible although the assertions are equivalent. Indeed, the execution of the agent in this platform will be rejected. Therefore, the integration of semantics and knowledge in the security domain at the intersection between policies seems to be very interesting. To solve this problem, we create an ontology in the mobile agent security domain to capture the following semantic information:

- XML-Encryption is a specification for encrypting/decrypting mobile agent XML data, this specification supports symmetric (3DES) and asymmetric (RSA) encryption algorithms.
- The X.509 certificate is an authentication mechanism, the mobile agent has a certificate containing his identity, a public key and data encrypted using his private key.

When this additional information is added to security policies, and semantic correspondence between policies is applied. Then, the comparator concludes that the capacity of the platform satisfies the agent's requirement and the platform's requirement is satisfied by the agent's capacity, making a perfect match between these two policies. This example illustrates the importance of semantic information to improve the quality of correspondence between security policies.

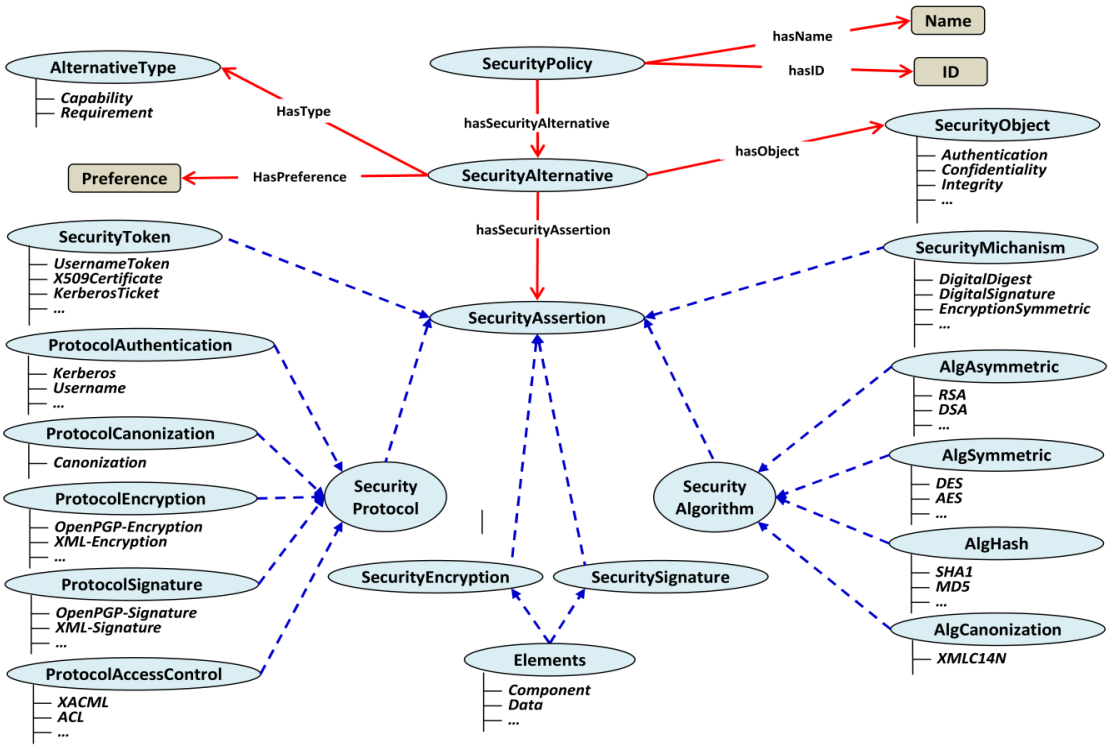

Fig.1. Main classes of security policy ontology

B. MASO Ontology to model security policies

A platform offers a set of services for mobile agents to perform their tasks. Each service has a set of functional properties. However, these properties are not sufficient to determine the most appropriate service for the agent's specific needs from a set of services that provide the same functionality. This is why it is important to have a clear description of its security policy, which allows the platform to express their requirements and security capabilities for each service [23]. In order to take security policies into account in the platform selection process for the secure execution of a mobile agent, we proposed an extension to the WS-Policy by adding new elements to its initial specification. These elements are expressed through safety concepts defined in the MASO ontology we have built.

This extension has allowed us to integrate the different security concepts, create semantic relations between these concepts and ensure automatic correspondence between security policies. We used a model based on an OWL ontology to represent these different elements.

Figure 1 presents the MASO ontology classes based on WS-Policy, to illustrate the difference between semantic relations and the class hierarchy. We use two lines to represent the relations between the different concepts of this ontology: The dotted line (blue color) links a specific class to a more general class, which allows defining the class hierarchies in the MASO ontology. The solid line (red color) allows you to specify the semantic relations between the different classes.

We create three classes SecurityPolicy, SecurityAlternative, and SecurityAssertion, in order to express security assertions within a security policy. Indeed, the SecurityPolicy concept is the top level class of our ontology. It represents the root of the security policy, each policy identified by a name and a unique identifier (Name, ID). It is consisted of at least one or more security alternatives (SecurityAlternative).

The SecurityAlternative class contains four semantic properties. The hasType property allows you to determine the type of the alternative with the AlternativeType class. This class contains two instances Capability and Requirement. The hasPreference property allows you to specify the preference of a particular alternative. The preference is expressed as xsd:int. The higher the preference value, the more weight the expressed preference has. If no preference is specified, the default value is zero. The hasObject property allows you to set the objective to be achieved by the SecurityObject class security alternative. Finally, the hasSecurityAssertion property allows you to specify the different security assertions used to satisfy a security objective. The SecurityAssertion class contains six subclasses (figure 1):

- SecurityMechanism describes the technical solutions and methods used to satisfy a security objective. This class has six instances: Authorization, DigitalSignature, DigitalDigest, EncryptionAsymmetric, EncryptionSymmetric and Identification
- SecurityProtocol allows specifying the different protocols and security specifications used to protect mobile agents and execution platforms
- SecurityAlgorithm contains the different algorithms for encryption, signature, hashing and data canonization. To do this, we have extracted four subclasses from this class. The AlgEncryption class has symmetric encryption algorithms to ensure data confidentiality. The AlgSignature class contains asymmetric encryption algorithms that ensure the authenticity and integrity of data. The AlgDigest class has the algorithms that allow you to create the data summary (MD5, SH1, SH2). The AlgCanonicalization class represents canonization algorithms, allowing XML information to be presented in a standard form.
- SecurityToken allows you to specify the different types of security tokens used by a security protocol or algorithm. Indeed, a security token can be used for authentication, encryption and data signing. There are six instances for this class: AsymmetricKey, SymmetricKey, SAMLAssertion, KerberosTicket, X509Certificate, and UsernameToken
- SecurityEncryption and SecuritySignature allow you to locate the elements to be encrypted/signed in the mobile agent. These two classes use the same Elements subclass to determine the elements to be protected. This class contains four instances: XPath, Data, Component, Itinerary.

C. MASO ownership constraint

We present the different semantic relations between security-related concepts in the mobile agent system, such as the security objective, security mechanism, protocols, algorithms, and others. Security policies will be defined on the basis of the MASO ontology. We redraw this ontology with new semantic properties (figure 2).

As shown in Figure 2, the SecurityObject class has several semantic properties to specify the security mechanisms, protocols, algorithms, and tokens that ensure a goal set by a security alternative. The ensuredByMechanism property allows expressing the security mechanisms used to satisfy a security objective. For example, the Confidentiality security objective is ensured by two security mechanisms EncryptionAsymmetric and EncryptionSymmetric. The other three semantic properties will be treated in the same way.

The supportProtocol property allows you to specify protocols that satisfy a security mechanism. For example, the XML-Signature protocol is used to guarantee the DigitalSignature security mechanism.

The adoptAlgorithm property allows a protocol to adopt one or more algorithms in its execution process. Some protocols require the presence of a security token with the required Token property. For example, the XML-Encryption protocol adopts the 3DES algorithm to

encrypt data and uses the SymmetricKey security token as an encryption key.

The usesToken property is used by the SecurityAlgorithm class to determine the list of keys used in the encryption/signature process (for example, a DSA signature algorithm uses an X509 security token to ensure the integrity and authenticity of the agent). The encryptedElement and signedElement property are used to determine the elements to be encrypted/signed of the mobile agent. The usesHash property is used between the AlgAsymmetric class and AlgHash to set the hash function adopted by the asymmetric algorithm. Finally, the usesCanonicalization property is used by the AlgAsymmetric class to specify the canonization algorithm to sign the mobile agent data.

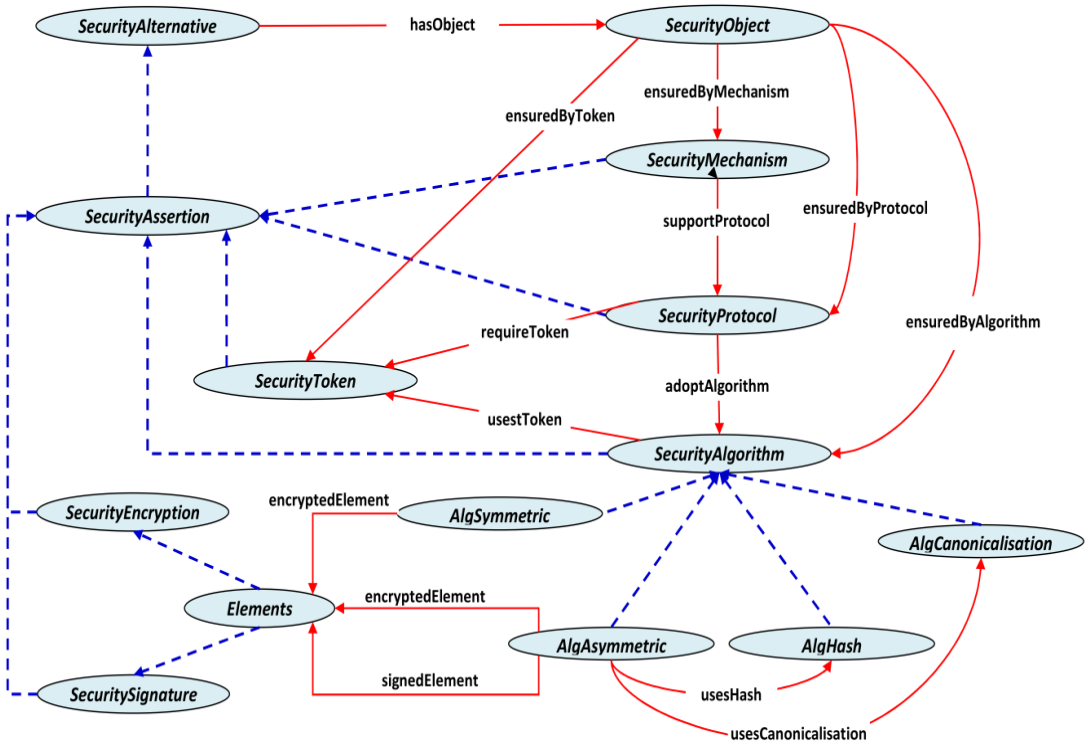

Fig.2. Semantic relationships between the different classes of MASO ontology

After defining all constraint properties and defining semantic relationships between classes. Our ontology for the mobile agent system becomes a universal way to express the security policy of the execution platform and the mobile agent.

D. Creation of security policies (requirements and capacities)

In our work, we have used the WS-Policy standard to express the requirements and capabilities within a security policy, based on the MASO ontology we have created. Indeed, security policies are expressed through the concepts defined by this ontology. They can be either instance of security protocols such as XACML, XML-Encryptions, or concrete security algorithms such as DES, RSA, or collections of instantiated features of these protocols such as confidentiality, authentication. In other words, safety requirements and capabilities can be described by using any component on an abstract level of the safety ontology. Each security policy can have more requirements and capacity.

Definition 1: We define a Security Requirement as a Requirement type security alternative, allows to achieve a specific security objective and to group together a set of MASO ontology security assertions to satisfy the objective. Formally, we have expressed the safety requirement (SR) by equation 1:

$$SR_{(object)} = \sum AS_R \quad (1)$$

AS_R is a set of safety assertions from the MASO ontology to express a safety requirement. For example, a mobile agent requires the confidentiality of the data generated by the execution platform, in which case the agent's policy requires an asymmetric RSA encryption algorithm (AS_1) with a security token X509Certificate (AS_2).

$$SR_{(confidentiality)} = AS_1 + AS_2$$

Definition 2: We also define a Security Capability as a Capability type security alternative, allowing us to offer a set of security mechanisms, protocols, and algorithms to achieve a particular security objective. Each alternative includes a set of safety assertions from the MASO ontology to meet the intended objective. Formally, we expressed security capacity (SC) by equation 2:

$$SR_{(object)} = \sum AS_C \quad (2)$$

AS_C is a set of MASO ontology safety assertions to express a safety capability. For example, a platform provides security capabilities to ensure data integrity; the platform policy provides an XML-Signature specification such as a signature protocol (AS_1) with the DSA algorithm (AS_2) and an X509 digital certificate (AS_3) to sign the mobile agent data.

$$SC_{(integrity)} = AS_1 + AS_2 + AS_3$$

The platforms specify their security requirements and capabilities in a policy that can be read by mobile agents. Also, agents have policies in place to express their security requirements and capabilities. The agent's security requirements must meet the platform's security features, so the platform's security requirements must also meet the security capabilities specified by the agent's policy. In the following, we present the semantic matching rules between requirements and security capabilities.

IV. Semantic Correspondences Between Security Policies

In this section, we will present the process of matching security policies based on the MASO ontology. Indeed, the process of assessing the correspondence between the two policies consists of seeking semantic compatibility between requirements and capabilities. In particular: (a) the platform requirements are compared with the capabilities of the mobile agent. (b) The platform capabilities are compared with the requirements of the agent. For this comparison to yield a positive result, the following two conditions must be met:

- The capabilities expressed in the platform's security policy must meet the requirements of the mobile agent
- The requirements of the platform must be respected by the capabilities expressed in the mobile agent security policy.

In the following, we detail the process of semantic correspondence between these two policies.

A. Policy Mapping Algorithm

We have developed a Matching-Algorithm to determine the level of correspondence between two security policies. Our algorithm accepts the agent's security policy and those of the platform as input and decides to what extent they match. This algorithm extracts the most specific type of requirement and capacity that ensure the same security objective and then checks its correspondence. The most specific type is the instance of the lowest class in the security ontology. We determine four possible matching results between a capacity and safety requirement:

Perfect-Match: A perfect match occurs when the requirement and capacity both refer to the same or two equivalent concepts. For example, if capacity and a requirement are both of the SAML type, then there is a perfect match between the requirement and the capacity. Generally, two cases are possible:

- Requirement and capacity refer to the same semantic notion
- Requirement and capacity refer to equivalent semantic concepts.

In both cases, if the properties of the requirement and the capacity are specified, then their values must be identical.

General Match: if the most specific type of capacity is lower in the hierarchy than the most specific type of requirement. In this case, it is said that the requirement is more general than the capacity. Three cases are possible:

- The requirement specifies a more general semantic concept than the capacity
- Requirement and capacity refer to the same semantic concept, but more details are specified for capacity (using the construction of the property)
- The requirement and capacity refer to the MASO safety ontology, but the requirement only specifies the safety objective, whereas the capacity is expressed by safety concepts that satisfy the objective specified by the requirement.

Negotiable-Match: if the most specific type of requirement is lower in the hierarchy than the most specific type of capacity. It is said that the capacity is more general than the requirement. In this case, the capacity does not adequately meet the safety requirement. For example, if the requirement is of type X509Certiificate, the capacity is of type Authentication. There are three possible cases:

- The requirement specifies a more specific semantic concept than the capacity
- Requirement and capacity refer to the same semantic concept, but the requirement specifies in more detail (using the construction of the property)
- Requirement and capacity refer to the MASO safety ontology, but capacity determines only one safety objective, while the requirement is expressed by safety concepts that satisfy the safety objective specified by the capacity.

No match (No-Match): if the most specific types of requirement and capacity have no relationship in the safety ontology, then there is no match between the two. Two cases are possible:

- Requirement and capacity refer to semantic concepts that have no semantic relationship
- The requirement and capacity refer to the same semantic concept, but their properties present

different specifications.

We have divided the process of semantic correspondence between policies into two steps. The first consists of determining the correspondence result between each pair of requirement and safety capacity, the objective is to find the capacity that best corresponds to a requirement. The second is the assessment of the overall correspondence between the two policies. The overall correspondence is defined as the minimum between the individual correspondence results evaluated in the first step.

Formally we have represented the semantic correspondence algorithm between two policies by a mathematical equation (3). We consider two security policies P1 and P2 (P1 for the agent and P2 for the platform). We define the $SR_{(object)}$ and $SC_{(object)}$ functions to express the security assertions of a requirement and a capability respectively.

$$\left.\begin{array}{l}(\forall R_i \in SR_{(object)}(p1) \exists C_j \in SC_{(object)}(p2) / C_j Satisfied R_i) \\ (\forall R_i \in SR_{(object)}(p2) \exists C_j \in SC_{(object)}(p1) / C_j Satisfied R_i)\end{array}\right\} \Rightarrow P1 match P2 \quad (3)$$

R_i and C_j represent the most specific requirement and capacity of Requirement and Capability safety alternatives to ensure a particular safety objective.

C_j Satisfied R_i means that the match result between C_j and R_i is Perfect-match or General-match.

B. Implementation of Matching-algorithm

The Integrated Development Environment (IDE) we must use must be extensible, universal, flexible, free and compatible with the chosen JADE platform. We chose ECLIPSE because it meets all the criteria listed. The specificity of Eclipse comes from its architecture totally developed around the notion of the plug-in: all the functionalities of this software workshop are developed as a plug-in.

To implement the semantic matching algorithm, we used two APIs to exploit and manipulate MASO ontology and mobile agent security policies and platforms:

Jena API to manipulate the MASO ontology. This API provides the basic level interface for RDF, RDFS and OWL files. It is a free software developed by HP's research laboratory in Bristol. Jena offers a set of SPARQL parsers and query engines in the form of Java classes.

JDOM API to manipulate agent security policies and platforms. This API allows you to analyze security policies, extract its different security concepts and represent these concepts in the form of a tree.

In the following, we present the class diagram of the semantic correspondence algorithm:

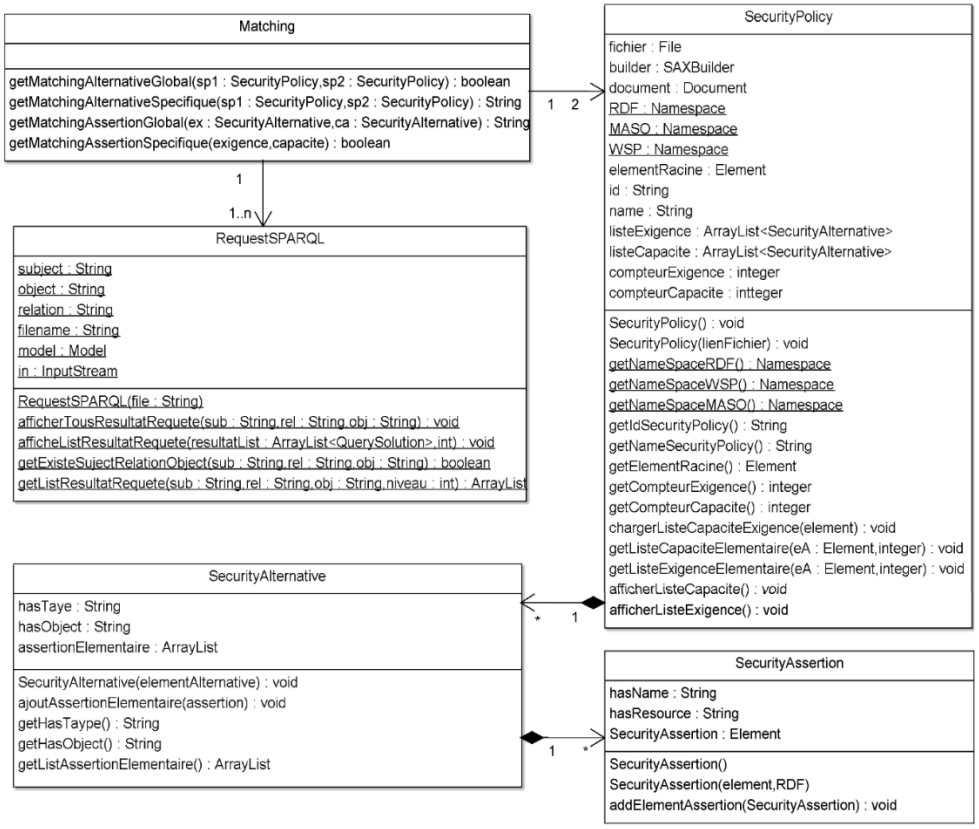

Fig.3. Class diagram of the semantic correspondence algorithm

The RequestSPARQL class allows you to query the MASO ontology with the SPARQL query language to retrieve the different relationships between a requirement and a security capability. Requirements and capabilities are retrieved from security policies (agent and platform) with the SecurityPolicy class.

The Matching class is used to check the semantic correspondence between two security policies (SecurityPolicy). It uses two types of verification. First, it verifies that each requirement in one policy has a security capability in the other policy ensures the same security objective. The second, it checks the individual correspondence between each requirement and safety capability pair, and then the overall correspondence between the two policies.

The SecurityPolicy class allows you to analyze the agent security policy and the platform. It retrieves the list of security requirements and capabilities for each policy, then the most specific security assertions for each requirement/capacity.

The SecurityAlternative class allows you to manipulate security alternatives, it retrieves the type, objective and list of security assertions used by an alternative.

The SecurityAssertion class allows you to manipulate the security assertions of each security alternative.

The matching algorithm accepts two security policies (agent and platform), then it gives the overall matching result between these two policies.

C. Example of semantic correspondence between two security policies

We have shown the importance of semantic correspondence between policies through the following example: a mobile agent has a security policy defined by its creator; this policy expresses the requirements that must be met by the platforms visited, as well as the security capabilities offered by the agent to the platforms. Figure 4 presents a description of the security policy used by the mobile agent.

```
1   <wsp:Policy
2           ID="502510120125" name="Mobile_Agent_1"
3           xmlns:wsp="http://www.w3.org/ns/ws-policy"
4           xmlns:MASO="http://namespace/MobileAgentSecurity">
5       <wsp:ExcactlyOne>
6           <MASO:SecurityAlternative>
7               <wsp:All>
8                   <MASO:hasType RDF:resource="&MASO;#Capability"/>
9                   <MASO:hasObject RDF:resource="&MASO;#Authentication"/>
10                  <MASO:SecurityAssertion>
11                      <MASO:ensuredByProtocol RDF:resource="&MASO;#X509"/>
12                      <MASO:supportProtocol RDF:resource="&MASO;#X509">
13                          <MASO:requireToken RDF:resource="&MASO;#X509Certificate"/>
14                      </MASO:supportProtocol>
15                  </MASO:SecurityAssertion>
16              </wsp:All>
17          </MASO:SecurityAlternative>
18      </wsp:ExcactlyOne>
19      <wsp:ExcactlyOne>
20          <MASO:SecurityAlternative>
21              <wsp:All>
22                  <MASO:hasType RDF:resource="&MASO;#Requirement"/>
23                  <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
24                  <MASO:SecurityAssertion>
25                      <MASO:ensuredByAlgorithm RDF:resource="&MASO;#RSA">
26                          <MASO:EncryptedElement RDF:resource="&MASO;#Data"/>
27                          <MASO:usesToken RDF:resource="&MASO;#X509Certificate"/>
28                      </MASO:ensuredByAlgorithm>
29                  </MASO:SecurityAssertion>
30              </wsp:All>
31          </MASO:SecurityAlternative>
32      </wsp:ExcactlyOne>
33  </wsp:Policy>
```

Fig.4. Mobile agent security policy

Figure 4 presents the security policy of the mobile agent. This agent has authentication capability with an X.509 digital certificate (line 8 to 15). Also, this agent requires in her security policy the encryption of the data produced by the platform visited with the asymmetric RSA algorithm (line 22 to 29). The prefix "MASO:" in the security policy indicates that these elements are referenced from the MASO security ontology.

It is assumed that a platform registered in an UDDI directory satisfies the functional needs of the mobile agent [23]. As illustrated in Figure 5, the platform requires in its security policy that visiting agents must authenticate their identity (line 7 and 8). Also, this platform supports the XML-Encryption encryption specification to encrypt/decrypt the data collected by the mobile agent (line 15 to 21).


```
1   <wsp:Policy ID="141200148566" name="Platform_1"
2       xmlns:wsp="http://www.w3.org/ns/ws-policy"
3       xmlns:MASO="http://namespace/MobileAgentSecurity">
4       <wsp:ExcactlyOne>
5           <MASO:SecurityAlternative>
6               <wsp:All>
7                   <MASO:hasType RDF:resource="&MASO;#Requirement"/>
8                   <MASO:hasObject RDF:resource="&MASO;#Authentication"/>
9               </wsp:All>
10          </MASO:SecurityAlternative>
11      </wsp:ExcactlyOne>
12      <wsp:ExcactlyOne>
13          <MASO:SecurityAlternative>
14              <wsp:All>
15                  <MASO:hasType RDF:resource="&MASO;#Capability"/>
16                  <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
17                  <MASO:SecurityAssertion>
18                      <MASO:ensuredByMechanism
                        RDF:resource="&MASO;#EncryptionAsymmetric">
19                          <MASO:supportProtocol RDF:resource="&MASO;#XML-Encryption"/>
20                      </MASO:ensuredByMechanism>
21                  </MASO:SecurityAssertion>
22              </wsp:All>
23          </MASO:SecurityAlternative>
24      </wsp:ExcactlyOne>
25  </wsp:Policy>
```

Fig.5. Security policy of an execution platform

The semantic matching algorithm decides whether the agent's security policy matches the platform's security policy. To do this, it accepts both policies as input and then makes the semantic correspondence. The following steps are taken by the algorithm to determine the degree of correspondence between these two policies:

Step 1: the algorithm analyzes the security policy of the mobile agent (Figure 4) and the security policy of the platform (Figure 5), then extracts the following semantic information:

For mobile agent (Figure 4):

- The mobile agent requires the confidentiality of the data produced by the platform visited (lines 22 and 23). To do this, it requires the use of the asymmetric RSA encryption algorithm (lines 25 and 28). In this case, the RSA algorithm is the most specific type for the confidentiality requirement
- The mobile agent offers authentication capability (lines 8 and 9), this authentication is based on the X.509 digital certificate (lines 11 to 14). In this case, the X509Certificate concept is the most specific type for authentication capability.

For platform (Figure 5):

- The platform requires authentication of mobile agents who want to use their services (lines 7 and 8). In this case, the Authentication objective is the most specific type for the authentication requirement
- The platform ensures the confidentiality of the data produced by the platform (lines 15 and 16). To do this, it has an encryption capability with the XML-Encryption specification (lines 18 to 20). In this case, XML-Encryption is the most specific type for confidentiality capability.

Step 2: the semantic matching algorithm determines the degree of match between the agent's requirement and the platform's capacity.

To ensure confidentiality, the agent requests in his security policy the use of the asymmetric RSA algorithm as a more specific requirement. The platform offers an XML-Encryption specification as a more specific capability. The requirement and capacity are not of the same type. In the MASO ontology, the XML- Encryption instance of the SecurityProtocol class allows adopting one or more encryption algorithms with the adoptAlgorithm property. In this case, XML-Encryption is a protocol that adopts the asymmetric RSA algorithm for asymmetric encryption. Indeed, the most specific type of requirement is lower in the hierarchy than the most specific type of capacity, so the degree of correspondence is Negotiable-Match.

Step 3: Similarly, the algorithm determines the degree of correspondence between the platform requirement and the agent's capacity.

To ensure the authentication objective, the mobile agent offers an authentication method with the X509Certificate as a more specific type. However, the platform requires the authentication of visiting mobile agents. In the MASO ontology, the X509Certificate instance is an instance of the SecurityToken class that ensures the authentication objective. In this case, the most specific type of capacity is lower in the hierarchy than the most specific type of requirement, so the degree of correspondence is General-Match.

Step 4: Finally, the semantic matching algorithm determines the degree of overall correspondence between the two policies. It takes the minimum of the degrees of correspondence between the individual matching results evaluated for each safety alternative. In this case, the

overall degree of correspondence is Negotiable-Match.

This example addresses the problem of semantic heterogeneity between the security policy of the mobile agent and the platform. However, the use of existing models based on syntactic approaches such as WS-Policy is not very well adapted to these heterogeneous and dynamic scenarios. To resolve this problem, we proposed an extension of WS-Policy with the MASO ontology for the creation of security annotations in mobile agent technology. It is much more comprehensive than security ontologies previously available in terms of the number of concepts, the properties of the concepts, and the type of resources that can be described. Its organization is also more intuitive so that it is easier to use as well as to extend New properties and instances can be added without modifying the overall class hierarchy. We demonstrated how ontology can be applied to the mobile agent system to describe capabilities and security requirements.

The algorithm we have developed is robust and powerful to facilitate automatic analysis of semantic compatibility between security policies. The algorithm takes into consideration not only concepts but also the properties of the concept. This is important because security annotations make extensive use of property attributes.

The creation of these ontologies is an iterative process. Additional instances and properties will always be needed to express new security statements. Classes and properties may be added and deleted as the security community continues to evaluate and refine the security ontologies. Further ontologies are still needed to address issues such as access control policies and quality of service in the mobile agent system. We hope this work will serve as a catalyst in the development of standardized security-related ontologies with contributions from both the security community and the semantic Web community.

V. Conclusion

We presented the modelling of security policies based on the WS-Policy specification. Then, we structured the security policy in terms of security requirements and capabilities. Then, we showed that syntactic correspondence poses a problem of incompatibility between security policies. To solve this problem, we have integrated the semantic and knowledge aspect in the security domain at the intersection between policies. For this purpose, an ontology has been developed in the field of mobile agent security "MASO".

The MASO ontology facilitates the automatic analysis of semantic compatibility between security policies, thus making it possible to annotate security capabilities and requirements with respect to various security concepts. We proposed an algorithm to make the semantic correspondence between the agent security policy and the platforms visited. The correspondence process consists of verifying the extent to which a security requirement is met by a security capability, the assessment of the correspondence between policies is based on the MASO ontology.

Reference

[1] P. Ahuja and V. Sharma, “A Review on Mobile Agent Security”, International Journal of Recent Technology and Engineering, 2012, vol. 1, pp. 2-5.

[2] H. Razouki and A. Hair, “Self-Adaptive Security for Mobiles Agents”, International Journal of Computer Applications, 2014, vol 94, n13, pp. 24-29.

[3] H. Razouki and A. Hair, “Towards A New Security Architecture of Mobile Agents”, International Journal of Soft Computing and Engineering, 2014, vol 3, n6, pp. 1-6.

[4] V. Singh and S.K. Pandey, “Revisiting security ontologies”, International Journal of Computer Science, 2014, vol 11, n 6, pp. 150-159.

[5] A. Musen Mark, “The Protégé project: a look back and a look forward”, AI Matters, 2015, vol 1, n°4, pp. 4-12.

[6] A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez and U. Yal¬cinalp, “Web Services policy 1.5 Framework”, W3C recommendation, 2007. http://www.w3.org/TR/ws-policy/.

[7] M.K. Srinivasan, K. Sarukesi, P. Rodrigues, M. SaiManoj and P. Revathy, “State-of-the art Cloud Computing security taxonomies: a classification of security challenges in the present Cloud Computing environment”, 2014, http://dl.acm.org/citation.cfm?id=2345474.

[8] B. Tsoumas and D. Gritzalis, “Towards an ontology based security management”, AINA 2TH International Conference Advanced Information Networking Applications, 2006, vol 1, pp. 985-992.

[9] I. Gorodetski, L.J. Popyack, I.V. Kotenko and V.A. Skormin, “Ontology-based multi-agent model of an information security system”, Proceedings of 7th International Workshop. RSFDGrC’99, 9-11 Nov 1999, pp. 528–532.

[10] A. Vorobiev and J. Han, “Security attack ontology for web services”, Second International Conference on Semantics Knowledge and Grid, 2006.

[11] S. Fenz, “Ontology-based generation of IT-security metrics. SAC”, 10 Proceeding of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1833-1839.

[12] J. Gao, B. Zhang, X. Chen and Z. Luo, “Ontology-based model of network and computer attacks for security assessment”, J. Shanghai Jiaotong Univ. (Sci.), 2013, vol 18, n5, pp. 554-562.

[13] A. Gyrard, C. Bonnet and K. Boudaoud, “The STAC (Security Toolbox: Attacks and Counter measures) ontology”, ACM Companion, 13–17 May 2013.

[14] A.A. Assali, D. Lenne and B. Debray, “Ontology development for industrial risk analysis”, IEEE International Conference on Information & Communication Technologies: from Theory to Applications (ICTTA 2008), Damascus, Syria, Apr 2008.

[15] S. Hacini and R. Lekhchine, “Security ontology for mobile agent’s protection”, International Journal of Computer Theory and Engineering, 2012, vol. 4, n3.

[16] S., Dritsas, L. Gymnopoulos, M Karyda., T. Balopoulos, S. Kokolakis, C. Lambrinoudakis and S. Katsikas, “knowledge-based approach to security requirements for e-health applications”, 2014. http://www.ejeta.org/specialOct06-issue/ejeta-special06oct-4.pdf.

[17] A. Evesti, R. Savola, E. Ovaska and J Kuusijärvi, “The design, instantiation, and usage of information security measuring ontology”, 2011.

[18] G. Elahi, E. Yu and N. Zannone, “A modeling ontology

for integrating vulnerabilities into security requirements conceptual foundations?", Proceedings of 28th International Conference on Conceptual Modeling, vol. 5829, 9–12 Nov 2009. Springer, Berlin Heidelberg, pp. 99-114.
[19] L. Obrsta, P. Chaseb and R. Markeloffa, "An ontology of the cyber security domain", http://ceurws.org/Vol966/STIDS2012_T06_ObrstEtAl_CyberOntology.pdf. Accessed 19 Feb 2014.
[20] T. Takahashi, Y. Kadobayashi, H. Fujiwara, "Ontological approach toward cyber security in Cloud Computing", Proceedings of the 3rd International Conference on Security of Information and Networks, 07 Sept 2010, pp. 100–109.
[21] E. Kamalakannan, B. Prabhakaran and K.S. Arvind, ". A study on security and ontology in cloud computing", International Journal of Advanced Research in Computer and Communication Engineering, 2013, vol 2, n10.
[22] A.M. Talib, R. Atan, R. Abdullah and M.A.M. Murad, Security ontology-driven multi-agent system architecture for cloud data storage security ontology development", International Journal of Computer Science and Network Security, 2012, vol 12, n5.
[23] H. Razouki and A. Hair, "Security for Mobile Agents: Trust Estimate for Platforms", TELKOMNIKA Indonesian Journal of Electrical Engineering, 2015, vol. 15, n2, pp. 381-389.

Authors' Profiles

Dr. H. Razouki completed his PhD in Computer Science at the University of Sultan Moulay Slimane, Morocco in 2017. Currently, he is an professor of computer science within the ministry of national education, professional formation, Higher Education and Scientific Research. His research interests are related to mobile agent system, and computer security. He has published more than 10 research papers at national and international journals, conference and proceedings.

How to cite this paper: Hassan RAZOUKI, "Security Policy Modelling in the Mobile Agent System", International Journal of Computer Network and Information Security(IJCNIS), Vol.11, No.10, pp.26-36, 2019. DOI: 10.5815/ijcnis.2019.10.04