I.J. Information Technology and Computer Science, **2016, 12, 67-74**
Published Online December 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2016.12.08

Improving Matching Web Service Security Policy Based on Semantics

Amira Abdelatey, Mohamed Elkawkagy, Ashraf Elsisi, Arabi Keshk
Faculty of Computers and Information/Computer Science, Menofia University, Egypt
E-mail: {Amira.mohamed, mohamed.elkhawaga, ashraf.elsisi, arabi.keshk}@ci.menofia.edu.eg.

***Abstract*—**Nowadays the trends of web is to become a collection of services that interoperate through the Internet. The first step towards this inter-operation is finding services that meet requester requirements; which is called a service discovery. Service discovery matches functional and non-functional properties of the requester with the provider. In this paper, an enhanced matching algorithm of Web Service Security Policy (WS-SP) is proposed to perform requirement-capability matchmaking of a consumer and a provider. Web service security policy specifies the security requirements or capabilities of a web service participant (a provider or a consumer). Security requirement or a capability of a participant is one of the non-functional properties of a web service. The security addressed through this paper is the integrity and the confidentiality of web service SOA message transmitted between participants. The enhanced matching algorithm states simple policy and complex policy cases of the web service security as a non-functional attribute. A generalization matching algorithm is introduced to get the best-matched web service provider from a list of available providers for serving the consumer.

Index Terms—Ontology matching, web service, SOA message security, Web service non-functional properties, web service security policy.

I. INTRODUCTION

Nowadays, web services become a modern standard in information technology industry. A service represents a self-contained platform-independent computational element that can be accessible by other applications across organizational boundaries [1]. In the service-oriented model, a service offered by a service provider and invoked by a service requester.

A discovery mechanism used by the provider to advertise its services and by the requester to find the suitable service that fulfills its requirements [2]. So, matchmaking and locating services is an important problem [3].Selecting web service must address not only the functional aspects but also non-functional properties of the service [4]. Web Service Description Language (WSDL) was inspired to represent the functional aspects of a Web Service. Web Service Policy (WS-Policy) used to represent the non-functional attributes of a web service. Both the functional and the non-functional capabilities and requirements of a Web Service is initial step during service discovery stage [5]. The discovery of web services is conducted by WSDL processing system [6]. That is besides the processing of WS-Policy.

This paper focuses on message security that is one of the non-functional properties of a web service [7]. Message security becomes a primary concern when using Web services. Message security mainly means the confidentiality and the integrity of data transmitted through the message [8]. Confidentiality and integrity are assured by applying security mechanisms such as encryption and digital signature. Framework implemented to perform automatic semantic matching between service requester's requirements and service provider's security capabilities. Different security classes are associated with web services like message encryption, digital signature, authentication, etc. [8]. Web Service Security Policy (WS-SP) specification used as a standard for representing security requirements for web service entities.

WS-Policy [9] is capable of representing the syntactic aspects of the non-functional properties but lacks semantics. It allows only syntactic matching of policies. It depends on intersection policy mechanism [10]. Syntactic matching of security policy restricts the effectiveness of checking the compatibility between requester and provider policies. As it has a strict yes-no matching result. Semantic matching leads to more flexible and correct result of matching policies. WS-SP transformed into The web Ontology Language Description Language (OWL-DL) [11]. Semantic Web Rule Language (SWRL) used for extending OWL-DL with semantic relations to get the best matching level between requester and provider policies. These relationships lead to more correct and more flexible matching of security policies. In this paper, an improved matching algorithm for WS-SP is introduced. It considers different cases of WS-SP types either simple or complex policy. A generalized matching algorithm is introduced for getting the best-matched provider from N number of providers.

In this paper, we will introduce an improved semantic matching algorithm of WS-SP. WS-SP is described in section II. Related work is discussed in Section III. In section IV, An improved WS-SP matching algorithm is presented. In addition, a generalization of the improved matching of WS-SP algorithm is provided in section V. The work concludes in Section VI.

II. Web Service Security Policy

Web service architecture as a type of SOA involves three entities: service provider, service registry and service consumer that integrate together to perform specific task [9]. In addition to WS-Policy, WS-SP (Web Service Security Policy) represents the syntactic aspects of security as a non-functional property. Because of the loosely coupled connections of SOA and HTTP as an open access, SOA must web services with a set of security requirements or capabilities.

Security is an important parameter of web service. Security refers the secure of SOAP message exchanged between provider and consumer. Message security assures SOAP message integrity and confidentiality, and identity. Each entity of a web service architecture has a requirement or a capability constraints. Matching these requirements or capabilities constraints is not an easy task.

WS-Policy allows Web services to define policy requirements for endpoints. These requirements include privacy rules, encryption rules, and security tokens. WS-SP allows Web services to apply security to Simple Object Access Protocol (SOAP) messages through encryption and integrity checks on all or part of the message.

WS-SP is an expressive language aggregated into the Web service architecture. Then matching WS-SP problem becomes more and more important while integrating Web services. However, WS-SP has a big weakness as it only allows syntactic matching of security policies. Security policy matching depends on the policy intersection mechanism provided by WS-Policy

A. *Security policy*

Security Policy is a widely spread in the industry, and it is currently a popular standard to be combined into the Web service architecture [12].Web Services Policy (WS-Policy) Framework is a framework for describing capabilities and requirements of web service provider and requester [10]. WS-Policy used to represent the non-functional properties of a web service. In the matching algorithm, the non-functional properties of a web service represent the policy requirements of the requester must be compatible with the capability policies of the provider. WS-SP used to specify the web service security specification for Web services.

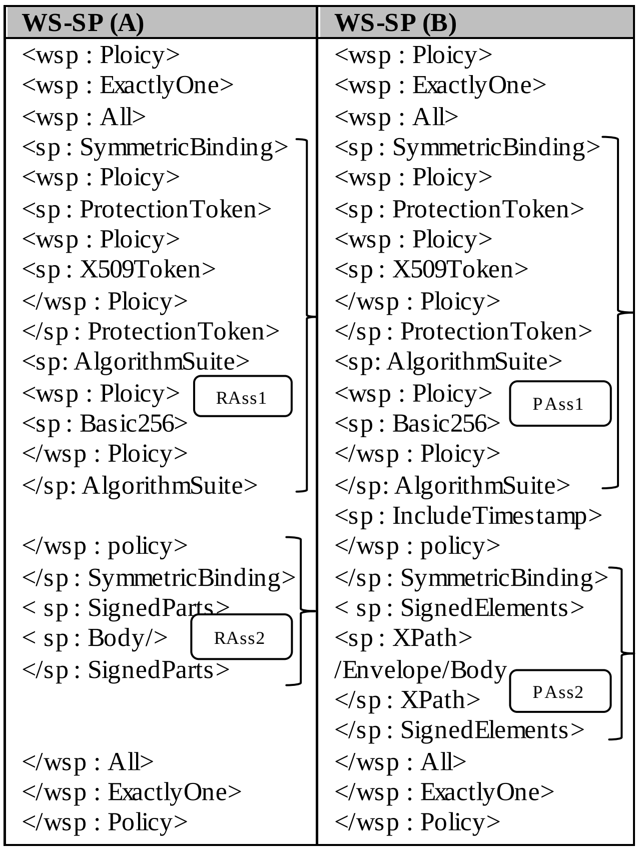
In WS-Security, a security policy defines a set of security policy assertions that used in determining individual security requirements or capability [13]. Policy operators used to combine security policy assertions. Policy operators have two elements: “Exactly One” and “Exactly All”. “Exactly one” used to express the assertions that have alternatives; it means only one of its children elements must hold. On the other hand, “Exactly All” means that all its children elements must hold. Alternatives used to describe requirement options of a requester or a provider.

Fig.1 shows security policy requirements are expressed using WS-SP. It represents security policy signature and encryption of a web service entity [14]. The example has one security alternative; this alternative have two assertions. This security policy supports the signature of the message body with a symmetric key securely transported using an X.509 token [15]. Besides, the necessary cryptographic operations must perform using Basic256 algorithm suite [16].

```
<wsp : Ploicy>
<wsp : ExactlyOne>
<wsp : All>
<sp : SymmetricBinding>
<wsp : Ploicy>
<sp : ProtectionToken>
<wsp : Ploicy>
<sp : X509Token>
</wsp : Ploicy>
</sp : ProtectionToken>
<sp: AlgorithmSuite>
<wsp : Ploicy>
<sp : Basic256>
</wsp : Ploicy>
</sp: AlgorithmSuite>
</wsp : policy>
</sp : SymmetricBinding>
< sp : SignedParts>
< sp : Body/>
</sp : SignedParts>
</wsp : All>
</wsp : ExactlyOne>
</wsp : Policy>
```

Fig.1. Representation Example of WS-SP

B. *Web service security policy matching problem*



WS-SP (A)	WS-SP (B)
<wsp : Ploicy>	<wsp : Ploicy>
<wsp : ExactlyOne>	<wsp : ExactlyOne>
<wsp : All>	<wsp : All>
<sp : SymmetricBinding>	<sp : SymmetricBinding>
<wsp : Ploicy>	<wsp : Ploicy>
<sp : ProtectionToken>	<sp : ProtectionToken>
<wsp : Ploicy>	<wsp : Ploicy>
<sp : X509Token>	<sp : X509Token>
</wsp : Ploicy>	</wsp : Ploicy>
</sp : ProtectionToken>	</sp : ProtectionToken>
<sp: AlgorithmSuite>	<sp: AlgorithmSuite>
<wsp : Ploicy> RAss1	<wsp : Ploicy> PAss1
<sp : Basic256>	<sp : Basic256>
</wsp : Ploicy>	</wsp : Ploicy>
</sp: AlgorithmSuite>	</sp: AlgorithmSuite>
	<sp : IncludeTimestamp>
</wsp : policy>	</wsp : policy>
</sp : SymmetricBinding>	</sp : SymmetricBinding>
< sp : SignedParts>	< sp : SignedElements>
< sp : Body/> RAss2	<sp : XPath>
</sp : SignedParts>	/Envelope/Body
	</sp : XPath> PAss2
	</sp : SignedElements>
</wsp : All>	</wsp : All>
</wsp : ExactlyOne>	</wsp : ExactlyOne>
</wsp : Policy>	</wsp : Policy>

Fig.2. Matching Problem of WS-SP

The automatic matching algorithm of WS-SP specifications checks requester’s policy against provider’s policy to ensure their compatibility. Syntactical matching of their policies is quite straightforward, but it lacks semantics [17]. It is not able to discover matching when each policy uses different vocabularies, even though they have the same meaning. It is necessary to construct a formal model that describes conceptual relationships between requester and provider policies. Ontology is the most commonly used formal, explicit specification of a shared conceptualization [18].

Fig.2 clarifies the matching problem of web service security policy.

WS-SP(A) and WS-SP(B) are two security policies each one has one alternative with two assertions. The assertions specified in the provider security policy ;WS-SP (B); are syntactically different from those specified in the requester security policy; WS-SP (A). So, policy intersection will adopt a “no match” result for these security policies. However, semantic analysis of the above provider security policy and requester security policy leads to a different matching result. Although RAss2 and PAss2 assertions don’t have the same type, they have the same meaning of signing the body of the message. The other difference between RAss1 and PAss1 assertions is that PAss1 includes an extra child element that is “sp: IncludeTimestamp” which means a timestamp element must be included in the security header of the message. From a security viewpoint, this strengthens the integrity of the service [19]. So, matching these two assertions must lead to a perfect match rather than a no match.

C. Policy matching

WS-Policy represents requester requirement and a provider capability. WS-Policy describes a normal policy form that is a disjunction of alternatives and conjunction of all assertions in an alternative. The proper form for policy matching is as follows: A policy P is defined as a finite set of alternatives *{Alt₁, Alt₂,….,Alt_N}*. It is expressed as a disjunction of all its alternatives as following:

$$\mathrm{P} = \mathrm{Alt}_1 \,|\, \mathrm{Alt}_2 \,|\, \ldots.\mathrm{Alt}_N \tag{1}$$

An alternative *Alt* is identified as a finite set of assertions *{Ass₁, Ass₂ ... Ass_N}*. It is also can be expressed as a conjunction of all its assertions as following:

$$Alt = Ass_1 \wedge Ass_2 \wedge Ass_N \tag{2}$$

The requester web service security policy *ReqP* is defined as a set of alternatives. Each alternative is a set of assertions. Requester policy expressed as follows:

$$\mathrm{ReqP} = \mathrm{Alt}_1 \,|\, \mathrm{Alt}_2 \,|\, \ldots.\ \mathrm{Alt}_i \tag{3}$$

$$Alt_i = Ass_1 \wedge Ass_2 \wedge Ass_i \tag{4}$$

Moreover, the provider web service security policy *ProvP* is defined as a set of alternatives. Each alternative is a set of assertions. The Provider policy expressed as follows:

$$\mathrm{ProvP} = \mathrm{Alt}_1 \,|\, \mathrm{Alt}_2 \,|\, \ldots.\mathrm{Alt}_j \tag{5}$$

$$Alt_j = Ass_1 \wedge Ass_2 \wedge Ass_j \tag{6}$$

Matching *ReqP* and *ProvP* security policies are reduced to finding equivalent alternatives. As expressed in the following rule.

$$((\exists Alt_i) S.T.\quad Alt_i \in \mathrm{Re}qP \quad and \quad (\exists Alt_j) S.T.\quad Alt_j \in \mathrm{Pr}ovP \quad and \quad Alt_i \Leftrightarrow Alt_j) \Rightarrow (\mathrm{Re}qP \Leftrightarrow \mathrm{Pr}ovP) \tag{7}$$

Finding equivalent alternatives is identified in the following manner. There are two alternatives are equivalent: if, for each assertion in both alternatives, there exists satisfied assertion. Equivalent alternatives expressed in the following rule.

$$((\forall Ass_i) S.T.\quad Ass_i \in Alt_i \quad and \quad (\exists Ass_j) S.T.\quad Ass_j \in Alt_j \quad and \quad Ass_i \in Ass_j) \Rightarrow (Alt_i \Leftrightarrow Alt_j) \tag{8}$$

From the above rules, an equivalent policy created from equivalent alternatives. Also, equivalent alternatives created from semantically equivalent assertions. In the proposed policy framework, equivalent assertions computed using semantic matching of these assertions.

D. Semantic relations of ws-sp

Semantic relations added to WS-SP ontology are defined in details in [20]. These semantic relations are prescribed by SWRL [21]. These relations bond the requestor SP and the provider SP. These rules define the conditions that requester and provider assertions must satisfy to create a given semantic relation. These relations express the semantic interpretation between requester and provider. First, from these semantic relations, we get the matching degree between assertions. After having the matching level of all assertions, we can get the matching level between policies.

E. WS-SP Ontology

The Ontology-based model of WS-SP consists of two main parts: an ontological representation of a Security Policy (SP) structure and an ontological representation of WS-SP assertions. Fig.3 shows the ontological representation of an SP structure. To specify SP in a normal form, there are three classes created “Security

Policy", "Security Alternative", and "Security Assertion". Security policy contains one or more alternatives. Security alternatives consist of one or more security assertion. Therefore, the three previous classes created in that particular order. In WS-SP standard, an assertion can have an arbitrary number of types: "Security Binding", "Protection Scope", "Supporting Security Tokens", "Token Referencing And Trust Options". These classes described in details in [20, 22].

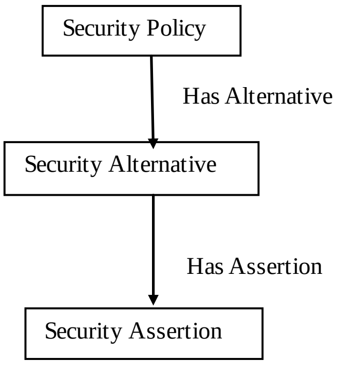


Fig.3. Ontological Representation of SP Structure

Security assertions classes modeled as ontological representation based on the semantic meaning of these assertions. "Security Binding", "Supporting Security Tokens", "Token Referencing And Trust Options", and "Protection Scope" are the security assertions of WS-SP ontology and are modeled as subclasses of the "Security Assertion" class as shown in Fig.4.

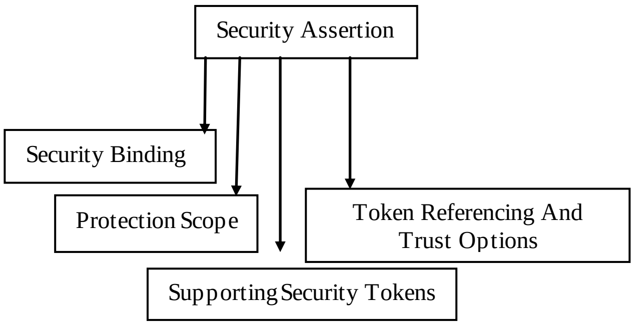

Security Assertion
Security Binding
Protection Scope
Token Referencing And Trust Options
Supporting Security Tokens

Fig.4. Class assertion types

The "Security Binding" class specifies the primary security mechanism to apply for securing message exchanges [23]. "Security Binding" can be either transport level, represented by "Transport Binding" class, or message level represented by "Message Security Binding" class. Transport binding protocols can be either symmetric binding and asymmetric binding which represented by the two subclasses "Symmetric Binding" and "Asymmetric Binding". "Protection Scope" class used to specify message parts encrypted and signed of security policy. It has two subclasses "Encryption Scope" and "Signature Scope". "Encryption Scope" class has two subclasses "Encrypted Element" and "Encrypted Part". "Signature Scope" class has two subclasses "Signed Element" and "Signed Part". "Supporting Security Tokens" class creates security binding elements and tokens. It supports tokens specify encryption and signing requirements. In other words, it supports security tokens required by the "Security Binding" class [24]. It has "Binary Security Token" class and "XML Security Token" class.

"Token Referencing And Trust Options" class defines various policy assertions related to exchanges between requester and provider. It is used for negotiation protocols. It has two subclasses "Trust Referencing Options" and "Trust Options".

III. RELATED WORK

Several works dealt with adding semantics to WS-SP to overcome the deficits of the policy intersection. M. Ben Brahim et al. [25] constructed a simple ontology to compare two security policies and then build an algorithm to compare security policies. This algorithm uses a semantic reasoner to get the result of the comparison. It uses WS-SP for specifying requester requirements and provider capabilities. It represents security requirements and capabilities as OWL ontology and reasoner works on top of it. The comparing algorithm is not described in detail.

S. Alhazbi et al. [26] introduced a framework for the preference based semantic matching between web services security policies. This approach utilizes the alternative feature in WS-policy to allow the requester to specify multi-optional requirements ranked by preference. Ontology used to model the relationships between different web service security concepts. The author also uses a reasoner to specify the level of matching. The matching algorithm uses a matching level and requester preference to specify the best option to be mapped with provider capabilities.

M. Ben Brahim et al. [22] presented a semantic matching technique to compare two security assertions. They proposed a WS-SP based ontology and some relations to compare two security assertions. They show how to get the matching level of two simple security assertions, but it lacks the comparing of all two policies. It also lacks processing of complex security policies.

T-D Cao et al. [20] presented a semantic approach for determining and matching the security policies. This approach transforms WS-SP into the OWL-DL ontology. It adds a set of semantic relations that can exist between the provider and requestor security concepts. The algorithm determines the matching level of the provider and requestor security policies. However, it lacks processing all probability cases of simple and complex security policies. It also lacks processing of complex policies.

The improved matching algorithm depends on [25], [22], [20]. It enhances the WS-SP based ontology semantic matching algorithm. It improves semantic matching simple security policy and complex security policy. In addition to considering all cases of a simple policy and complex one.

IV. Improved Semantic Matching Algorithm

Web service security policy matches the compatibility of WS-SP alternatives and assertions. Improved matching algorithm used for matching two security policies. The matching process checks to what extent requester security requirements satisfied by provider capabilities. It first checks whether the security policy is a simple policy or a complex one. The simple policy is a security policy that has only one alternative with any number of assertions. The complex policy is a security policy that has more than one alternative each alternative has one or more assertions.

Work [25] has discussed matching security policies. After that, they extended their work to discuss matching provider and requestor security policies in a simple policy case and complex policies cases [20]. They do not clarify how to apply the matching algorithm in complex policies. In addition, in contrast, all the standards [10, 17, 27] say alternative has more than one assertion. The authors in the [20] apply their matching with considering that a policy contains different assertions and an assertion contains one or more alternative.

We categorize WS-SP as a simple policy and a complex policy. As stated before, the simple policy is a policy which each web service entity has zero or one alternative. If requester alternatives and provider alternatives equal zero, then a perfect match. If one of requester alternatives and provider alternatives equal zero and the other has one or more alternative, then no match. If requester alternatives and provider alternatives equal one, then simple policy matching. If each of provider alternatives or requester alternatives has one alternative and the other has more than one alternative, then complex policy matching. If provider alternatives and requester alternatives have more than one alternatives, then complex policy matching. Different cases for simple policy matching and complex policy matching stated in Table 1.

Table 1. Simple policy and complex policy different cases

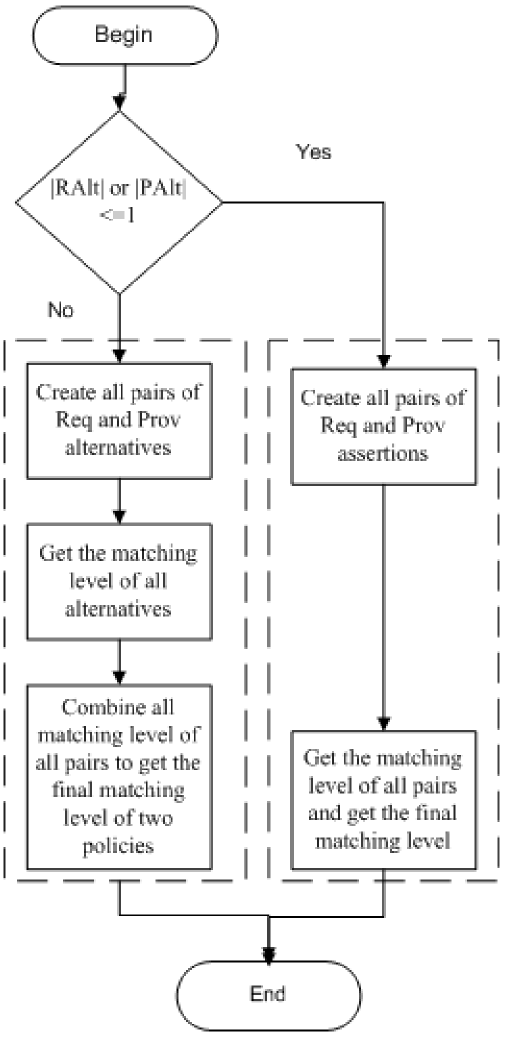
Requester	Provider	State
|RAlt|=1 or |RAlt| > 1	|PAlt| = 0	(Simple Policy) No Match
|RAlt| = 0	|PAlt| = 0	(Simple Policy) Perfect Match
|RAlt| = 0	|PAlt|=1 or |PAlt| > 1	(Simple Policy) No Match
|RAlt| = 1	|PAlt| = 1	Simple Policy
|RAlt| >1	|PAlt| = 1	Complex Policy
|RAlt| = 1	|PAlt| > 1	Complex Policy
|RAlt| > 1	|PAlt| > 1	Complex Policy

Therefore, the Simple matching process conducted in only one case "if requester and provider have one alternative". The complex matching process carried out if requester and provider have one or more alternatives. Previous works on this topic do not consider all different cases. The improved algorithm studies all possible cases.

There are four possible assertion-matching levels for requester and provider assertions: perfect match, close match, possible match and no match. These assertion-

```
compareAssertion(ReqAss,ProvAss)
{
If  ( ReqAss "isIdenticalTo" ProvAss )
    then, perfect match.
If ( ReqAss "isEquivalentTo" ProvAss )
    then, perfect match.
If( ReqAss "isMoreSpecificThan" ProvAss )
    then, close match.
If ( ReqAss "isMoreGeneralThan" ProvAss )
    then, possible match.
If ( ReqAss "isLargerThan" ProvAss )
    then, possible match.
If  ( ReqAss "isStrongerThan" ProvAss )
    then, possible match.
If (ReqAss "hasTechDiffWith" ProvAss )
    then, possible match.
If  ( ReqAss "isDifferentFrom" ProvAss )
    then, no match.
If  ( ReqAss "isSmallerThan" ProvAss )
    then, no match.
If  ( ReqAss "isWeakerThan" ProvAss )
    then, no match.
}
```

Fig.5. compareAssertion () pseudo code



Begin
|RAlt| or |PAlt| <=1
Yes
No
Create all pairs of Req and Prov alternatives
Get the matching level of all alternatives
Combine all matching level of all pairs to get the final matching level of two policies
Create all pairs of Req and Prov assertions
Get the matching level of all pairs and get the final matching level
End

Fig.6. Improved matching security policy algorithm

matching levels depend on previously described semantic relations. Assertion matching levels described in details in [25]. CompareAssertion () procedure compares requester assertions with provider assertions. It returns perfect match, possible match, close match and no match. The pseudo code of compareAssertion () is shown in Fig.5.

If requester assertion has identical relation or equivalent to the relation with provider assertion, then a perfect match. If requester assertion has been more specific than relation with provider assertion, and then close match. If requester assertion has been more general than or is larger than or is stronger than or has tech diff with relations with provider assertion, then possible match. If requester assertion has been different from or is smaller than or is weaker than relation with provider assertion, then no match.

The matching process starts with the requirement of a Requester, so the requester is defined as the starter of matchmaking process [13]. Matching algorithm mainly depends on a number of alternatives and number of assertions that are the primary components of the security policy. To get the final matching level of a simple policy, get the final assertion matching level as the lowest degree of the match found between requester assertions and provider assertions. In complex policy, matchmaker matches alternatives of requester security policy with provider alternatives. To get the final matching level of complex policy, get the final matching level as the highest level of the match found between requester and provider alternatives. Alternative matchmaker calls assertions matchmaker, which is the simple policy case. The improved matching security policy algorithm is shown in Fig.6.

Through Fig.6, it first checks the number of requester and provider alternatives that are the primary component of policy. If requester alternatives "RAlt" or provider alternatives "PAlt" less than or equal to "one" then simple policy else complex policy. Through a simple policy, it creates all pairs of requester and provider assertions. Then, get the matching level of all pairs and finally get the matching level with the lowest value of matching. It gets the matching level of each pair by calling "compareAssertions" pseudo code.

With the complex policy, the algorithm first creates all pairs of the requester and provider alternatives. Second, it gets the matching level of all alternatives. and finally, it combines all matching level of all pairs and gets the final matching level of two policies by finding the highest matched alternatives. During complex policy matching process, the algorithm calls simple policy matching. The complexity of policy is analyzed by comparing the elements in Requester Policy P1 with a number of alternatives X1, the number of Requester assertions X2 and Provider Policy P2 with a number of alternatives Y1, the number of Provider assertions Y2.

Table 2 defines a complexity for the improved matching algorithm compared to [25] and [20].

Table 2. Complexity of improved matching algorithm

Requester	Provider	Work [25]	Work [20]	Improved matching algorithm
|RAlt|=1 or |RAlt| > 1	|PAlt| = 0	▨	▨	0
|RAlt| = 0	|PAlt| = 0	▨	▨	0
|RAlt| = 0	|PAlt|=1 or |PAlt| > 1	▨	▨	0
|RAlt| = 1	|PAlt| = 1	O(X2.Y2)	O(X2.Y2)	O(X2.Y2)
|RAlt| >1	|PAlt| = 1	▨	▨	O(X1.X2.Y2)
|RAlt| = 1	|PAlt| > 1	▨	▨	O(Y1.X2.Y2)
|RAlt| > 1	|PAlt| > 1	▨	▨	O(X1.Y1.X2.Y2)

Complexity for a simple policy of the improved matching is the same as the time in [20, 25] , which take into account only the number of assertions of requester and provider. The complexity of complex policy is defined in the improved matching algorithm only. Note that the gray cells represent the simple and complex policies that are not considered in [20, 25].

V. Generalization of Improved Matching Algorithm

The improved semantic matching of WS-SP matches requester security policy with provider security policy. It only matches one requester with one provider and returns perfect match, possible match, close match and no match. Through the web service selection phase, Requester selects the best-matched provider. Therefore, a requester matches a number of providers.

As a generalization of matching WS-SP, the requester matches with N number of providers to get the most suitable provider. After processing of the generalized matching algorithm, the requester chooses the best-matched provider of the N numbers.

Table 3. Complexity analysis of a generalized matching

Requester	Provider	Improved matching algorithm
|RAlt|=1 or |RAlt| > 1	|PAlt| = 0	0
|RAlt| = 0	|PAlt| = 0	0
|RAlt| = 0	|PAlt|=1 or |PAlt| > 1	0
|RAlt| = 1	|PAlt| = 1	N*O(X2.Y2)
|RAlt| >1	|PAlt| = 1	N*O(X1.X2.Y2)
|RAlt| = 1	|PAlt| > 1	N*O(Y1.X2.Y2)
|RAlt| > 1	|PAlt| > 1	N*O(X1.Y1.X2.Y2)

Table 3 defines the complexity of the generalized matching between one requester and N number of providers. For the increased time, a parallel technique used to decrease the processing time for the generalized matching WS-SP algorithm. Processing of matching the requester with each provider executed separately.

VI. Conclusion and Future Work

In this paper, an improved web service security policy-matching algorithm is introduced with considering all cases of simple security policy and complex security policy. In this paper, we considered the complex policy cases of WS-SP matching. In addition, we state all cases of simple and complex policies. In simple policy, we state all instances of simple policy. Furthermore, the improved algorithm addressed the complex policy with all different situations. A generalized matching of WS-SP is conducted to get the best-matched provider from the different set of providers.

As a future work, we aim to extend the improved algorithm to match requester security requirements with different provider security policies and get the best-matched provider. Also, we target to add a negotiation technique so that interaction between web service provider and the consumer can take place.

References

[1] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, 2003, pp. 3-12.

[2] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *Services Computing, IEEE Transactions on*, vol. 3, pp. 223-235, 2010.

[3] A. FELLAH, M. Malki, and A. ELÇI, "Web Services Matchmaking Based on a Partial Ontology Alignment," 2016.

[4] E. M. Maximilien and M. P. Singh, "Toward autonomic web services trust and selection," in *Proceedings of the 2nd international conference on Service oriented computing*, 2004, pp. 212-221.

[5] T. Lavarack and M. Coetzee, "Considering web services security policy compatibility," in *Information Security for South Africa (ISSA), 2010*, 2010, pp. 1-8.

[6] N. N. Chiplunkar and A. Kumar, "Dynamic Discovery of Web Services using WSDL," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 6, p. 56, 2014.

[7] L. Seinturier, P. Merle, R. Rouvoy, D. Romero, V. Schiavoni, and J. B. Stefani, "A component - based middleware platform for reconfigurable service - oriented architectures," *Software: Practice and Experience*, vol. 42, pp. 559-583, 2012.

[8] D. Jamil and H. Zaki, "Security issues in cloud computing and countermeasures," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, pp. 2672-2676, 2011.

[9] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*: Prentice Hall PTR, 2005.

[10] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, *et al.*, "Web services policy 1.5-framework," *W3C Recommendation*, vol. 4, pp. 1-41, 2007.

[11] J. De Bruijn, R. Lara, A. Polleres, and D. Fensel, "OWL DL vs. OWL flight: conceptual modeling and reasoning for the semantic Web," in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 623-632.

[12] P. Hallam-Baker, V. M. Hondo, H. Maruyama, M. McIntosh, and I. Nataraj Nagaratnam, "Web Services Security Policy Language (WS-SecurityPolicy)," 2005.

[13] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *International Semantic Web Conference*, 2003, pp. 402-418.

[14] J. H. An, Y. Dodis, and T. Rabin, "On the security of joint signature and encryption," in *Advances in Cryptology—EUROCRYPT 2002*, 2002, pp. 83-107.

[15] C. Adams and D. Pinkas, "Internet X. 509 public key infrastructure time-stamp protocol (TSP)," 2001.

[16] H. V. Chung, Y. Nakamura, and F. Satoh, "Security Policy Validation For Web Services," ed: Google Patents, 2007.

[17] S. Speiser, "Semantic annotations for ws-policy," in *Web Services (ICWS), 2010 IEEE International Conference on*, 2010, pp. 449-456.

[18] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, *et al.*, "Bringing semantics to web services: The OWL-S approach," in *Semantic Web Services and Web Process Composition*, ed: Springer, 2005, pp. 26-42.

[19] K. Ono, Y. Nakamura, F. Satoh, and T. Tateishi, "Verifying the consistency of security policies by abstracting into security types," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*, 2007, pp. 497-504.

[20] T.-D. Cao and N.-B. Tran, "Enhance Matching Web Service Security Policies with Semantic," in *Knowledge and Systems Engineering*, ed: Springer, 2014, pp. 213-224.

[21] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML," *W3C Member submission*, vol. 21, p. 79, 2004.

[22] M. B. Brahim, T. Chaari, M. B. Jemaa, and M. Jmaiel, "Semantic matching of ws-securitypolicy assertions," in *Service-Oriented Computing-ICSOC 2011 Workshops*, 2012, pp. 114-130.

[23] N. Gruschka and L. L. Iacono, "Vulnerable cloud: Soap message security validation revisited," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 625-631.

[24] D. Z. G. Garcia and M. B. F. De Toledo, "Ontology-based security policies for supporting the management of web service business processes," in *Semantic Computing, 2008 IEEE International Conference on*, 2008, pp. 331-338.

[25] M. Ben Brahim, T. Chaari, M. Ben Jemaa, and M. Jmaiel, "Semantic matching of web services security policies," in *Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on*, 2012, pp. 1-8.

[26] S. Alhazbi, K. M. Khan, and A. Erradi, "Preference-based semantic matching of web service security policies," in *2013 World Congress on Computer and Information Technology (WCCIT)*, 2013.

[27] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, *et al.*, "WS-SecurityPolicy 1.3," *OASIS Standard, February*, pp. 41-44, 2009.

Authors' Profiles

Arabi keshk received the B.Sc. in Electronic Engineering and M.Sc. in Computer Science and Engineering from Menoufia University, Faculty of Electronic Engineering in 1987 and 1995, respectively and received his PhD in Electronic Engineering from Osaka University, Japan in 2001. His research interest includes software testing, software engineering, distributed system, database, data mining, and bioinformatics.

Ashraf El-Sisi received the B.Sc. and M.Sc. in Electronic Engineering and Computer Science Engineering from Menofia University, Faculty of Electronic in 1989 and 1995, respectively and received his PhD in Computer Engineering & Control from Zagazig University, Faculty of Engineering in 2001. His research interest includes cloud computing, privacy preserving data mining, and Intelligent systems.

Mohamed Elkawkagy, (1973) ,male, Faculty of Computers and Information, Menofia University, Egypt, Lecturer, received his PhD in 2012, his research directions include AI-planning, Software Engineering ,Planning search strategy , Multi-agent Planning, Web-based planning, Agent Systems and Human Computer Interaction (HCI)

Amira Abdelatey received the B.Sc. and M.Sc. in computers and information from Menofia University, Faculty of computers and information in 2007 and 2012, respectively. Currently hold PhD student in Faculty of computers and information, Menofia University. Her research interest includes semantic web, web service, intelligent systems, web service security, software engineering and database systems.

How to cite this paper: Amira Abdelatey, Mohamed Elkawkagy, Ashraf Elsisi, Arabi Keshk, "Improving Matching Web Service Security Policy Based on Semantics ", International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.12, pp.67-74, 2016. DOI: 10.5815/ijitcs.2016.12.08