Considering web services security policy compatibility

Tristan Lavarack[1] and Marijke Coetzee[2]
Academy for Information Technology
University of Johannesburg
Gauteng, South Africa
tlavarack@gmail.com[1], marijkec@uj.ac.za[2]

Abstract— For most organizations supporting business-to-business (B2B) web services interactions, security is a growing concern. Web services providers and consumers document their primary and alternative security policy requirements and capabilities in security policy files, defined by WS-Policy, WS-SecurityPolicy and WS-Security syntax. To secure message exchanges to the satisfaction of all parties, the security requirements of both web services providers and consumers need to be satisfied. This paper investigates how mutually agreed-upon security policies can be created. An analysis of the policy intersection algorithm highlights its deficiencies for finding mutually compatible policies. The interrelated effect that security policy assertion choices have on each other is identified as an important aspect not yet considered. Over and above security policy assertions, other influence on security policy choices, which may affect the security level supported by the organization, is identified. A proposal is made on how the assertions of two security policies should be considered, in order to create a secure, mutually agreed-upon security policy that will satisfy the requirements of both parties.

Keywords: WS-Policy; WS-SecurityPolicy; policy intersection, security policy assertions, policy compatibility

I. INTRODUCTION

For the last decade, e-business has been on the rise, supported by web services technologies. Businesses experience the benefits that web services technology provides as they extend their business processes beyond the physical boundaries of their enterprise [1]. As the future vision of the Internet of Services (IoS) [20] is realized, security requirements of B2B web services are becoming increasing important to address. In IoS, services need to interact continuously, across domains and even international borders. To enable secure interoperation for first generation web services, security policies are defined by means of metadata associated with services, and exchanged by proprietary protocols.

Web service providers specify their security requirements and capabilities in machine-readable security policies that web services consumers have to conform to. If they cannot conform, they need to search for other web services providers with whom they can find security policy compatibility. For web services consumers that already have their own set of security policies and mechanisms in place, this proves a difficult problem. In this modern day, web services providers thus need to be more flexible and accommodate a variety of security requirements.

Currently, the compatibility between the security policies of web services consumers and providers is determined by policy intersection. Policy intersection suffers from a number of limitations as semantic meaning of policy assertions is not considered. It is thus an inadequate method to solve this problem.

To harness the flexibility needed in managing the security of a web service, new approaches are required, where security policies of both web services providers and consumers are carefully considered to create a security policy acceptable to both parties. As the underlying platforms hosting B2B web service interactions are becoming more complex, where hosts are networked in complicated topologies using firewalls and intrusion detection systems, the definition of an adequate security policy is no easy task. The configuration of non-functional aspects such as security thus requires a deep understanding by administrators.

This paper gives an analysis of security policy intersection. Currently, WS-Policy [3] and related WS-SecurityPolicy [12] specifications determine policy compatibility by using policy intersection. The main contribution of this paper is to give an overview of the limitations of policy intersection, and to highlight additional considerations that should be taken into account when mutually compatible policies are defined. Section II begins the paper by reviewing web services and their corresponding security specifications using an example to highlight important aspects. Section III analyses the intersection of two security policies. Section IV gives a high-level model of aspect to consider and section V concludes the paper.

II. BACKGOUND

Security for web services is implemented is a unique way. The WS-Policy specification [3] defines an XML Schema that defines the main structure of a security policy. Specifications such as WS-Security [2] further define platform independent, domain-specific security mechanisms that are used to specify the rules of the security policy. Web services and security specifications namely WS-Security, WS-Policy and WS-SecurityPolicy [12] are briefly discussed next.

A. Web services

Web services are exposed to consumers via the use of explicitly defined interfaces, documented in Web Services Definition Language (WSDL) [4] files. WSDL files define the functional characteristics of web services such as location and structure of messages. The WSDL syntax cannot be used to specify non-functional characteristics for web services such as quality-of-service or security [5]. To specify such non-functional requirements, additional policies are needed. For security, there are many policy specifications that can additionally be used such as WS-Policy [3], WSPL [6] and WS-Agreement [7].

To better understand the security requirements of web service interactions, consider the following example: WebSupply is a web service provider supporting the sale of digital media such as music Compact Discs (CDs) to retail stores. WebSupply provides services that allow online customers of web services consumers to search a selection of music on CD's, order and pay for the ordered CD's. As confidential information is exposed such as client information and credit card numbers, SOAP messages need to be well-protected. As a consumer, NewHitsMusic is a retail CD store that has recently discovered WebSupply as provider of services. NewHitsMusic requires the integration of the services of WebSupply into their application environment to the benefit of their online customers. NewHitsMusic is a new company and have not yet managed to support a comprehensive number of security mechanisms. When credit card numbers are exchanged, they have to be protected by security mechanisms such as encryption algorithms that both parties can successfully apply to a part of the SOAP message.

Next, security policy specifications, used to specify the security policies of Web Supply and NewHitsMusic are discussed. In order to be able to specify security mechanisms, WS-Security is discussed next.

B. WS-Security

As web services messages have the ability to pass through untrusted or unknown intermediaries before reaching their destination, point-to-point security mechanisms such as HTTPS are not sufficient. WS-Security provides end-to-end security for web services by extending SOAP to include security mechanisms, such as Kerberos, XML signature and XML encryption, to create a framework to imbed security in a SOAP message, in a transport-neutral way [11]. WS-Security specifies three main mechanisms:

- Security tokens for authentication,
- Encryption of SOAP messages for confidentiality,
- Signing of SOAP messages for integrity and non-repudiation.

WS-Security uses a variety of signature formats, encryption algorithms and authentication tokens [12]. To imbed security into a SOAP message, a security header is added to it as shown in Figure 1. The `<wsse:UsernameToken>` tag contains all information needed to send a secure message with a username, password and timestamp. The username tag, `<wsse:Username>` requires a plain text username; "Bob" and the `<wsse:Password Type="wsse:PasswordDigest>"` tag is the password digest. Because a password digest is required, a nonce is used inside the `<wsse:Nonce>` tags. A timestamp is inserted between the `<wsu:Created>` tags to give the SOAP message additional protection.

```
<Wsse:UsernameToken>
    <wsse:Username>Bob</wsse: Username>
        <wsse:Password Type="wsse:PasswordDigest">
            Pea-s=s!w@o$r(d
         </wsse:Password>
    <wsse:Nonce>abc123</wsse:Nonce>
    <wsu:Created xmlns:wsu="http://schemas.xmlsoap.org
    /ws/2002/07/utility">
        2010-04-08T13:30:30Z
    </wsu:Created>
</wsse:UsernameToken>
```

Figure 1: WS-Security header with Username Token

WS-Security only specifies platform-independent security mechanisms. In order to allow a consumer to understand how to format a security header, and to explicitly state all other security requirements and capabilities, a security policy needs to be defined.

C. WS-Policy

WS-Policy is a framework for defining XML based policies and has been standardized in 2007. Policies consist of policy assertions that represent domain-specific capabilities, constraints or requirements [13], specified by for example, WS-Security and WS-SecurityPolicy. Policy assertions are grouped together to form a policy expression. Each policy has a subject such as a web service port, operation or message to which the policy can be bound.

Figure 2 is an example of a policy defined in WS-Policy. The policy uses three operators to control assertions namely; `<wsp:Policy>`, `<wsp:All>`, and `<wsp:ExactlyOne>` [13]. Namespaces `wsu`, `sp` and `wsp` represent the WS-Services Utility, WS-Security and WS-SecurityPolicy namespaces respectively. `<wsp:Policy>` is a container for nested policy assertions. Each `<wsp:Policy>` has a unique ID value by use of the `wsu:id` attribute. The `<wsp:All>` operator requires that all child assertions contained within it are satisfied.

```
<wsp:Policy wsu:Id="WebSupplyPolicy">
   <wsp:All>
      <wsp:ExactlyOne>
         <sp:UsernameToken>...</sp:UsernameToken>
         <sp:X509Token>...</sp:X509Token>
      </wsp:ExactlyOne>
      <sp:IncludeTimestamp/>
   </wsp:All>
</wsp:Policy>
```

Figure 2: WebSupply security policy.

As Figure 2 contains one `<wsp:All>` tag, all the child assertions have to be satisfied. The policy has two nested assertions. The first assertion, surrounded by a `<wsp:ExactlyOne>` tag, has two nested assertions relating to the use security tokens. The second assertion is compulsory and requires the use of time stamps. The policy thus states with the `<wsp:ExactlyOne>` operator that either a UsernameToken or a X509Token must be used. A timestamp is always needed. The definition of policy alternatives is discussed next.

1) Policy Alternatives

WS-Policy allows the use of policy alternatives [3] to allow the specification of policy choices. Policy alternatives are the building blocks of combined security policies. By being able to specify alternatives in a policy, security policies become less static in nature as web services consumers are given a choice between security requirements of web services providers. With more, equally secure, sets of security requirements and capabilities, a web services provider will thus be able to interact with more diverse web services consumers.

Before policy alternatives can be compared with each other, the policy has to be in normal form to clarify the content of all alternatives [15]. Normal form of policies is a standardized format where only one `<wsp:ExactlyOne>` operator is used. `<wsp:All>` tags are used to represent policy alternatives, which are nested in the `<wsp:ExactlyOne>` tag. Figure 2 is converted into normal form, and shown in Figure 3. Each `<wsp:All>` tag contains a set of nested assertions There are two alternatives, the one requires the use of a Username Token and timestamp, and the other requires the use of a X.509 Certificate Token and timestamp.

```
<wsp:Policy>
   <wsp:ExactlyOne>
      <wsp:All>
         <sp:X509Token>...</sp:X509Token>
         <sp:IncludeTimestamp/>
      </wsp:All>
      <wsp:All>
         <sp:UsernameToken>...</sp:UsernameToken>
        <sp:IncludeTimestamp/>
      </wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 3. Part of WebSupply's security policy in normal form.

WS-Policy provides the structure and rules of policy processing. If developers were to be given a free hand when designing security policies, much confusion will arise. To ensure that standard, usable security policies are defined, WS-SecurityPolicy is used.

D. WS-SecurityPolicy

The WS-SecurityPolicy [12] defines a set of policy assertions that are used to define individual security requirements or constraints of a web service. It reuses the operator set defined in WS-Policy to create security policies that contain policy alternatives with nested security assertions.

The range and structure of security aspects over which compatibility between the service consumer and provider must be reached is defined by these security specifications. ISO 7498-2 [19] defines 5 main categories of security services namely authentication, access control, confidentiality, integrity and non-repudiation. The main focus of WS-SecurityPolicy is on authentication, confidentiality and integrity. Mechanisms for non-repudiation are not explicit, but can be applied with integrity and binding mechanisms. Access control is either left to the web services provider to implement, or can be defined with SAML or Kerberos Tokens. WS-SecurityPolicy incorporates WS-Security to define policies that can use weaker security mechanisms such as the transport security provided by HTTP, or much stronger security mechanisms such as a custom combination of XML signature and encryption. Administrators need to carefully evaluate chosen mechanisms and their combinations in order to determine the strength of security that is supported by a security policy. There are five main policy assertion types:

- *Token assertions* specify security tokens such as X509 certificates that provide public/private keys when a SOAP message is signed and encrypted.
- *Security binding assertions* define the way in which SOAP message exchanges are secured, such as the use of HTTPS transport protection when the Transport binding assertion is selected.
- *Protection assertions* specify which message parts are protected and how they are protected for selective signing and encryption of SOAP message parts.
- *Supporting token assertions* specify security tokens used to provide additional claims about a message sender such as security tokens used in authentication.
- *Protocol assertions* are used to specify predefined security requirements for SOAP message security and trust related options that SOAP message senders and receivers must both support. For example, the Wss10 assertion requires that the sender and receiver are able to process external URI references.

The Token assertions and Security binding assertion are now further examined as they provide the foundation for a security policy.

The *Token Assertion* is used to specify the types of tokens used for SOAP message protection such as `Username` Tokens, `X509` Tokens, `SAML` Tokens and `HTTPS` Tokens. The second part of Table 1 gives supported authentication tokens. If stronger authentication tokens are used, better identification and trust in the other party is possible.

The *Security Binding Assertion* defines the process used to secure SOAP message exchanges [12]. Three binding assertions are defined by WS-SecurityPolicy namely the *Transport* binding assertion, *Asymmetric* binding assertion and the *Symmetric* binding assertion. For Transport binding, SOAP message security point-to-point security is provided. The SOAP message sender and receiver have a restricted level of security as they may, for example, not be able to specify which message parts need to be signed, thereby lowering the level of security provided.

Table 1: Algorithm suites and authentication tokens

Confidentiality and Integrity							
Algorithm Suite	[Dig]	[Enc]	[Sym KW]	[Asym KW]	[Enc KD]	[Sig KD]	[Min SKL]
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192
Basic256Sha256	Sha256	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192Sha256	Sha256	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128Sha256	Sha256	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDesSha256	Sha256	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Sha256Rsa15	Sha256	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Sha256Rsa15	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Sha256Rsa15	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesSha256Rsa15	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

Authentication Token
HttpsToken
UsernameToken
IssuedToken
SpnegoContextToken
SecurityContextToken
SecureConversationToken
RelToken
SamlToken
KerberosToken
X509Token

By applying message protection at the SOAP encoding layer instead of at the transport layer, more flexible security policies at finer level of granularity can be defined with Asymmetric Binding and Symmetric Binding for multi-tier architectures. These bindings use security tokens and keys to selectively sign and encrypt parts of SOAP messages to provide end-to-end security as messages moves across domains. This allows for flexible control over confidentiality and integrity that is not present in Transport binding.

Symmetric binding is generally used in situations where only the web services provider has an X.509 certificate [12]. A common security token is used for SOAP message exchange. A symmetric key is created and encrypted using the keys derived from the security token and used for all message encryption and signature operations. The symmetric key is encrypted using the public key of the web services provider and is sent with the SOAP message. If both parties possess X.509 certificates, Asymmetric binding is recommended.

In Asymmetric binding two unique security tokens are used, provided by each party [16]. The public-private key pairs used for signing and encryption are derived from X509 certificates or SAML tokens. The private keys are used to sign a SOAP message and the public keys are used to encrypt a SOAP message.

For confidentiality and integrity, a sound collection of cryptographic algorithms, listed in Table 1, is defined by an algorithm suite, for performing operations such as signing, encryption, and generating message digests.

For example, `Basic256`, listed in the first row, incorporates the `AES256` encryption algorithm, `Sha1` hash function, `KwAes256` key wrap algorithm for symmetric keys, `KwRsaOaep` key wrap algorithms for asymmetric keys, `PSha1L256` encryption key derivation algorithm and `PSha1L192` signature key derivation algorithm and `256` minimum key length.

The choice of algorithm has an influence on the strength of message security. For example, the choice of digest influences the strength of integrity, and the choice of encryption algorithm has an influence on the strength of confidentiality. The key wrap algorithms influence strengths of both integrity and confidentiality. The strengths of encryption algorithms can be ordered from strongest to weakest as `AES-256`, `AES-192`, `AES-128`, and `Triple-DES`. Similarly, each of the columns can be ordered from strongest to weakest e.g. `TripleDesSha256Rsa15` is the weakest algorithm suite.

Finally, additional features such as `TimeStamps` and `MessageIDs` can be used to further assist with non-repudiation and protect against replay attacks. The WS-SecurityPolicy specification is complex and addresses a large variety of additional aspects not discussed here such as the order of encryption and hashing, and whether both the header and body of the message must be protected. Administrators need to carefully evaluate chosen mechanisms and their combinations in order to determine the strength of security that is supported by a security policy. When two policies are intersected, it would be important to ensure that the list of compatible alternatives provide a sufficient level of security to both the web service provider and consumer.

E. Policy Intersection

Policy intersection finds the matching alternatives of two policies by using an intersection algorithm [13]. Policy intersection is a commutative and associative function that takes two policies as input and returns a policy containing the compatible alternatives. If two policy alternatives are compatible, their intersection is an alternative containing all of the assertions found in both alternatives. If the alternatives that are being combined do not agree on the same vocabulary, they are not added to the new policy. For example, if a web service provider's security policy requires authentication with certificates and a consumer uses username-password combinations, no compatibility between the policies can be found.

The intersection algorithm consists of two steps namely domain-independent policy intersection and domain-specific processing. The WS-Policy specification does not explain how domain-specific processing should be implemented. This is left to the individual or organization in charge of the domain-specific processing.

For domain-independent policy intersection, two policies in normal form, with a set of policy alternatives and their nested assertions have to be present. Policy intersection is implemented as follows. Policy alternatives from both policies are compared to each other using the following rules. If two policy assertions have the same type, they are compatible. The type of an assertion is specified by the Qualified Name (QName) property of an assertion. The QName is unique and identifies what an assertion does. For example, `sp:ProtectionToken` specify protection tokens that need to be present in a respective policy alternative. If an assertion has a nested policy with alternatives, it is only compatible with an assertion that has a nested policy with compatible assertions [3].

Once policy intersection has been applied to two policies, all compatible policy alternatives discovered during policy intersection are included in a new policy. For two incompatible policies, policy intersection will result in an empty policy with no matching assertions. The new policy can then be processed or used as necessary. To demonstrate policy intersection, an example is now examined.

```
<wsp:Policy>
   <wsp:ExactlyOne>
   ---<wsp:All>
   |     <sp:SymmetricBinding>                    [1]
   |        <sp:ProtectionToken>                  [2]
   |           <sp:X509Token>...</sp:X509Token>
   |        </sp:ProtectionToken>
  (A)       <sp:AlgorithmSuite>                   [3]
   |           <sp:Basic192/>
   |        </sp:AlgorithmSuite>
   |        <sp:IncludeTimestamp/>                [4]
   |     </sp:SymmetricBinding>
   ---</wsp:All>

   ---<wsp:All>
   |     <sp:TransportBinding>                    [5]
   |        <sp:ProtectionToken>                  [6]
   |           <sp:HttpsToken>...</sp:HttpsToken>
   |        </sp:ProtectionToken>
   |        <sp:AlgorithmSuite>                   [7]
  (B)          <sp:Basic256/>
   |        </sp:AlgorithmSuite>
   |        <sp:SupportingTokens>                 [8]
   |           <sp:UsernameToken/>
   |        </sp:SupportingTokens>
   |     </sp:TransportBinding>
   ---</wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 4: WebSupply's security policy in normal form.

Figure 4 gives a simple security policy for the service provider, WebSupply. It extends the policy defined in Figure 3 with more features using WS-SecurityPolicy syntax.

In this policy, the first policy alternative (A) requires the use of symmetric binding with timestamps. `Basic192` is required as the algorithm suite from which the signing and encrypting algorithms are defined.

The second policy alternative (B) requires the use of transport level security provided by HTTPS. The `Basic256` algorithm suite is required as well as a `Username` token to authenticate the sender of the message.

These two options of varying strength and complexity provide web services consumers with a choice of security policy alternatives to choose from. The policy is in normal form, which means that it is ready for policy intersection.

Figure 5 shows NewHitsMusic, a web services consumer's security policy. NewHitsMusic does not support a sophisticated platform and support more basic security mechanisms. A weaker security token namely `Username` token, with a weaker algorithm suite namely `Basic128` is used over transport binding.

```
<wsp:Policy>
   <wsp:ExactlyOne>
   ---<wsp:All>
   |     <sp:TransportBinding>                    [9]
   |        <sp:ProtectionToken>                  [10]
   |           <sp:HttpsToken>...</sp:HttpsToken>
   |        </sp:ProtectionToken>
   |        <sp:AlgorithmSuite>                   [11]
  (C)          <sp:Basic128/>
   |        </sp:AlgorithmSuite>
   |        <sp:SupportingTokens>                 [12]
   |           <sp:UsernameToken/>
   |        </sp:SupportingTokens>
   |     </sp:TransportBinding>
   ---</wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 5. NewHitsMusic security policy.

NewHitsMusic determines whether it can support the security mechanisms of WebSupply by first performing policy intersection over the two policies defined in Figure 4 and Figure 5. Policy alternative B from the WebSupply security policy in Figure 4 and policy alternative C defined in the NewHitsMusic security policy from Figure 5 match as these alternatives have the same number and type of nested assertions. Assertions 5 to 8 from alternative B in Figure 4 match the assertions 9 to 12 from alternative C Figure 5. All of the matching assertions are included in a new security policy shown in Figure 6.

```
<wsp:Policy>
   <wsp:ExactlyOne>
   ---<wsp:All>
   |     <sp:TransportBinding>                    [13]
   |        <sp:ProtectionToken>                  [14]
   |           <sp:HttpsToken>...</sp:HttpsToken>
   |        </sp:ProtectionToken>
   |        <sp:AlgorithmSuite>                   [15]
   |           <sp:Basic256/>
   |        </sp:AlgorithmSuite>
   |        <sp:SupportingTokens>                 [16]
   |           <sp:UsernameToken/>
  (D)       </sp:SupportingTokens>
   |        <sp:ProtectionToken>                  [17]
   |           <sp:HttpsToken>...</sp:HttpsToken>
   |        </sp:ProtectionToken>
   |        <sp:AlgorithmSuite>                   [18]
   |           <sp:Basic128/>
   |        </sp:AlgorithmSuite>
   |        <sp:SupportingTokens>                 [19]
   |           <sp:UsernameToken/>
   |        </sp:SupportingTokens>
   |     </sp:TransportBinding>
   ---</wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 6.WebSupply and NewHitsMusic intersected security policy.

After policy intersection, the policy in Figure 6 contains duplicates and inconsistencies. In alternative D, assertions 15 and 18 both specify an algorithm suite of different strengths. There are also two duplicate `Username` token assertions. This policy will confuse developers using it and it needs to be corrected with domain-specific processing.

In the next section, policy intersection is evaluated to identify aspects that need to be addressed to ensure that a resultant policy does not lead to a less secure environment.

III. POLICY INTERSECTION EVALUATION

Policy intersection is by itself not intended to create correct, useable security policies. WS-Policy intersection is solely focused on syntactic of alternatives and does not address the semantics of assertions, or their influences on each other [10]. Subtle differences between assertions cannot be managed properly. There is also no guidance on how to address domain-specific processing in a standard manner. Additional policy processing is required to correct policies after policy intersection, resulting in a two step policy intersection process. Also, as NewHitsMusic generally supports weaker security mechanisms, policy intersection ensures that the agreed upon policy includes these mechanisms, without considering how their interdependence will affect the strength of security supported by WebSupply.

To highlight the limitations of policy intersection, domain-independent policy intersection, related influence of policy alternatives and assertions, and the influence of external factors on security policy intersection are now discussed.

A. Domain-independent policy processing

The manner in which policy assertions are constructed and the absence of semantic matching of assertions are limitations that lead to inconsistent policies when policy intersection is performed.

- *Policy inconsistencies:* When security policies are intersected, the new security policy will contain all the matching assertions which can create semantic inconsistencies in the form of assertion duplication or contradicting assertions. For duplicate assertions, the policy intersection algorithm does not interpret if assertions have the same type and if so, whether the same underling mechanism is specified. If assertions are exactly the same, only one copy of the assertion should be placed in the new security policy. For contradicting assertions of the same type such as assertions 15 and 18 of alternative D in Figure 6 both specify an algorithm suite, but with different strength namely `Basic128` and `Basic256`. The policy intersection algorithm cannot decide which one is best to use and an out-of-band discussions between administrators of the two environments is needed.
- *Assertion incompatibility:* Policy intersection only considers assertions to be compatible if they share the same type. If two assertions are slightly different they will not match. Take for example the two assertions in figure 7, that are very similar. They both require that some form of a supporting token has to be used. The only difference is that the first assertion requires a time stamp to be used, while the second one does not. In policy intersection these assertions will not intersect. An intelligent intersection mechanism should be able to detect that semi-compatible assertions such as these are similar enough to be placed in a policy.

```
//Assertion 1
<wsp:All>
   <sp:SupportingTokens>...</sp:SupportingTokens>
   <sp:IncludeTimestamp/>
</wsp:All>

//Assertion 2
<wsp:All>
   <sp:SupportingTokens>...</sp:SupportingTokens>
</wsp:All>
```

Figure 7. Two similar security assertions

- *Assertion parameters:* Attributes and child elements of assertions are completely ignored by policy intersection, thereby not detecting incompatibilities.

WS-policy does not address how domain-specific policy processing should be implemented. The next sections highlight some considerations that need to be taken into account by domain-specific policy processing namely the related influence of security mechanisms and of external influences.

B. Related influence of security mechanisms

Security policy alternatives and assertions specified in security policies are typically defined in isolation from each other. However, the usefulness of any mechanism often lies with the way in which it is combined with others. It is thus important for security policy administrators to view their security mechanisms in the context of the whole security system. It would also be important for security policy administrators to be able to evaluate alternative security mechanisms against each other to determine which will be the best for the given situation. This is even more important when policy intersection has been performed.

The security mechanisms used in a security policy alternative all contribute to the security level of a security policy. Stronger security mechanisms help to increase the security level while weaker security mechanisms lower the security level. To create an appropriate security policy alternative, a mix of stronger and weaker security mechanisms can be used to reach a certain security level. For example, in Figure 4, WebSupply uses transport binding with a Username token, but requires a strong algorithm suite to protect messages. Unfortunately, policy intersection does not consider the contribution that each security mechanism makes towards reaching a specific security level.

The security level or security goal of a web service is directly affected by integrity, confidentiality and authentication mechanisms used. This research considers the related effects that security mechanisms and policy assertions for these security services may have on each other and on the security level of the organization.

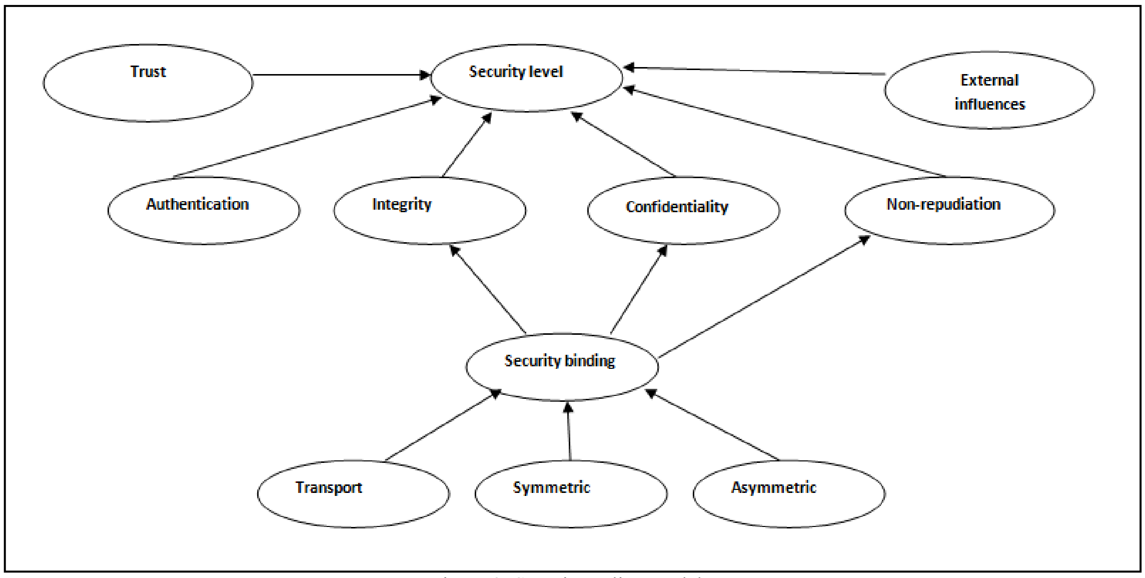

Figure 8: Security policy model

Authentication is an important security service to implement. If a web services consumer cannot be properly identified, the web services provider will not provide services to the web services consumer. Because of this, authentication has a strong influence on the security level. The security mechanisms used for authentication must therefore carefully be selected and protected. Authentication has a direct influence on the trust relationship with the other party. On the other hand, if there is high trust in the other party, the weaker authentication tokens may be used. With low trust between two parties, stronger forms of authentication tokens have to be used to strongly identify each party to each other.

An important influence on integrity and confidentiality is the security binding, as it defines a set of properties that together give coherent information on how to secure a given message exchange. For example, one can stipulate that an asymmetric token is used with a digital signature to provide integrity protection, and parts of a message are encrypted with a symmetric key which is then encrypted using the public key of the recipient. The security binding restricts what can be placed in the security header of a message and the associated processing rules. A decrease in either of the strength of confidentiality and integrity mechanisms will negatively influence the security binding. The security binding is also influenced by the choice of algorithm suite, the binding type and the use of timestamps. By using a strong algorithm suite, the security level supported by the security binding will be improved as it ensures a sound combination of security mechanisms for integrity and confidentiality. The type of binding such as Asymmetric binding can ensure more fine-grained message security, as parts of a message can be protected as it moves across domains. If Transport binding, is used, HTTPS and not SOAP security is applied, providing point-to-point protection, of lesser strength. Including timestamps strengthens integrity, confidentiality and provides non-repudiation evidence.

Current policy intersection processing does not address any of these complexities. In the next section, the external influences to policy intersection are discussed.

C. *External influences*

Computing environments supporting web services applications are becoming more complex and diverse, as complicated network topologies using firewalls, intrusion detection systems and intermediate proxy servers are created. If an organisation's environmental scanners detect a heightened number of attacks on the organisation's systems, it would require of consumers to use better confidentiality and integrity mechanisms to counter this danger. The selection of policy alternatives thus dictates a profound understanding of the complexities of the environment and their influences on each other. External influences are specific to the web service provider or consumer environment, and influence the choice of policy alternatives directly. For example, vulnerabilities scanners or firewalls, metrics collected when security mechanisms are used, and trust mangers that monitor the trust level between the negotiating parties can be considered. These influences differ according to the circumstances and preferences of each provider or consumer. For example, a SME may have very different security preferences, influences on its security level, and security goals than a large enterprise.

IV. SECURITY POLICY FRAMEWORK

Currently, security policy intersection is very limited. Compatible security policies may present risks to organizations as the combination of security alternatives may include inconsistencies and errors. In order to comprehensively consider all important aspect when security policies are intersected, a first step towards a security policy model is presented in Figure 8. It presents a high-level view of the relationships between the different aspects that were discussed. .

Figure 8 indicates which security mechanisms and policy assertions influence each other. The security mechanisms, such as specific algorithms resort under each respective component.

When policy intersection is performed, a trade-off analysis is required between policy assertions in compatible security policies to ensure that the best set of policy assertions to use. There are a number of steps required in this process.

1. Identify the security preferences of the environment as well as security mechanisms that have been implemented.
2. Assign a weight to each security mechanism to be able to determine which are preferable to use.
3. Determine a security goal that a security policy should support.
4. Once issues have been identified, decision-making mechanisms must be employed to understand the impact of choices and resolve disputes. Use intelligent decision-making to select the best policy alternatives. The very nature of such decisions is a fuzzy and uncertain process that is domain and context dependent. In a cooperative process of negotiation, consumers and providers are more likely to be satisfied with the final result if they participated in reaching the result by way of compromises and trade-offs.

V. Conclusion

In this paper, the need to find mutually compatible security policies was identified. Web services security policy specifications were discussed using an example. The policy intersection algorithm provided by WS-Policy was analyzed and a number of weaknesses associated with security policy intersection were identified. An important contribution made was the discussion the inter-related effect that the selection of security mechanisms has on each other, and on the security level supported by the security policy.

The focus of future research is to design a tool to support the features that were identified by this research. It will be of great assistance to administrators to have a graphical interface to view the influences that security policy selection has on the security level supported by the policy in conjunction with external influences.

References

[1] M.P. Papazoglou and P. M. A. Ribbers, “e-Business: Organizational and Technical Foundations”, Wiley, 2006.

[2] WS-Security, http://www.oasis-open.org/committees/wss.

[3] Boubez, T., Hirsch, F., Hondo, M., Orchard, D., Vedamuthu, A., Yalcinalp, U., Yendluri, P.:WS-Policy, http://www.w3.org/TR/ws-policy/. Web Services Policy 1.5 – Framework (2007)

[4] M. Duran and J. Hasan, “Expert Service-Oriented Architecture in C# 2005”, Second Edition, Apress, USA, p. 15, 2006.

[5] L. Baresi, S. Guinea, and P. Plebani, “WS-Policy for Service Monitoring”, 6th International Workshop on Technologies for E-Services (TES 2005, Trondheim, Norway), Lecture Notes in Computer Science (LNCS), Vol. 3811. Springer, pp. 72-83, 2005.

[6] A. Anderson, “An Introduction to the Web Services Policy Language (WSPL)”, IEEE 5th International Workshop on Policies for Distributed Systems and Networks, New York, USA, 7-9 June, 2004.

[7] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke and M. Xu, Web Services Agreement Specification(WS-Agreement), Grid Resource Allocation Agreement Protocol (GRAAP) Working Group, 2007.

[8] Cardoso. J, Voigt. K, Winkler. M, “Service Engineering for the Internet of Services, Enterprise Information Systems”, LNBIP, Vol.19, pp.15-27, Springer, 2009.

[9] K. Scarfone, A. Singhal, T. Winograd, “Guide to Secure Web Services”, Recommendations of the National Institute of Standards and Technology, August, 2007.

[10] Hudert. S, Eymann. T, Ludwig. H, and Wirtz. G, “A Negotiation Protocol Description Language for Automated Service Level Agreement Negotiations”, In: 2009 IEEE Conference on Commerce and Enterprise Computing, Washington, DC, pp. 162-169, 2009.

[11] S. Seely, “Understanding WS-Security”, Technical Articles, Microsoft Corporation, October, Available: http://msdn.microsoft.com/en-us/library/ms977327.aspx, 2002.

[12] N. Mihindukulasooriya, “Understanding WS – Security Policy Language”, WSO2 Inc., 28 January, Available: http://wso2.org/library/3132, 2008.

[13] B. Hollunder, “Domain-Specific Processing of Policies or: WS-Policy Intersection Revisited”, 2009 IEEE International Conference on Web Services, Los Angeles, USA, 6-10 July, 2009.

[14] Barbir, A., Goodner, M., Granqvist, H., Gudgin, M., Nadalin, A.: WS-SecurityPolicy 1.2. OASIS Standard, 1 July (2007)

[15] P. Nolan, “Understand WS-Policy processing”, IBM, 7 December, Available: http://www.ibm.com/developerworks/webservices/library/ws-policy.html, 2004.

[16] T. Nurmela, “WS-Policy specifications”, MOBILE WEB SERVICES, 2005.

[17] WS-SecurityPolicy Examples, http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html.

[18] I. Suriarachchi, “WS-Security Processing Models Along with WS-SecurityPolicy”, WSO2 Inc., 22 September, Available: http://wso2.org/library/articles/ws-security-processing-models-along-ws-securitypolicy-1, 2008.

[19] International Organization for Standardization, “IS0 7498-2, Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture”.

[20] Schroth, C.: The Internet of Services: Global Industrialization of Information Intensive Services, In: Proceedings of the 2nd IEEE International Conference on Digital Information Management, Web X.0 and Web Mining Workshop, IEEE Computer Society, Lyon, France, (2007)