A Framework for Protecting Web Sevices with IPsec

Cássio Ditzel Kropiwiec, Edgard Jamhour, Carlos Maziero
PPGIA – Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba – PR – Brazil
{cassio,jamhour,maziero}@ppgia.pucpr.br

Abstract

This paper proposes a WSDL security extension for describing IPsec policies in Web Services negotiations. The proposed extension is based on the policy model proposed by IETF for describing IPsec policies. Besides the model, the paper presents a framework for embedding IPsec negotiations in Web Services applications. The paper also presents the results of the overhead evaluation introduced by the use IPsec to protect Web Services applications under different configurations.

1. Introduction

This paper proposes a mechanism for using IPsec (Internet Protocol Security) in order to protect the communication between *Web Services* and its clients. According to our proposal, a *Web Service* (server) informs to its clients, along with the offered service descriptions, a set of IPsec policies that must be used in order to have access to its services. The IPsec policies, written in XML, are embedded into the WSDL (Web Services Description Language) files that describe the offered services. A client application uses the WSDL policy information for configuring its local host with a compatible set of IPsec policies necessary for having access to the service. The mechanism proposed in this paper defines a framework that automates the process of configuring IPsec policies in the client, interpreting directly the WSDL file. The paper shows how that, by using XML transformations, the selected policies can be automatically converted into IPsec commands, compatible with the client's operating system.

The contributions of this paper can be summarized as follows. First, the paper presents a XML scheme for mapping the standard IPsec policy model, proposed by IETF, into XML files [10]. There is no IETF publication concerning XML mapping of policy based information (the existent publications concerns mapping policy information into LDAP schemes). Second, the paper defines the procedure for embedding the XML policies into WSDL files. Third, the paper defines a framework for automatically interpreting and transforming policies into commands to the host operating system of the client. Because the security policies of a client can be more restricting than those proposed by the server, the framework also defines a procedure permitting a client to select only the policies that accommodate its "minimum requirement policy specification". And finally, the paper presents the analysis impact of using IPsec for protecting Web Services communications under several IPsec configurations. This study has been implemented using the framework described in this paper, and the IPsec configurations have been described using the proposed WSDL extension.

This paper is structured as follows: section 2 presents an analysis of the security mechanisms that can be used to protected Web Services applications and presents the reasons that motivated the use of IPsec in this work. Section 3 presents the proposed XML model for representing IPsec policies. Section 4 presents the proposed framework for automating the process of interpreting and enforcing IPsec policies defined using the proposed model. Section 5 presents a case study that evaluates de proposal and measures the performance impact of the use of IPsec in Web Services calls under different configurations. Finally, the conclusion synthesizes the important issues of this work, and points for future developments.

This paper assumes the reader is familiar with basic Web Services concepts, and its elements: SOAP (Simple Object Access Protocol) [4] and WSDL (Web Services Description Language) [5]. This paper also assumes that the reader is familiar with IPsec and its protocols and key negotiation mechanisms (IKE, Internet Key Exchange). For these topics, please refer to the IPsec IETF working group [9].

2. Security Mechanisms for Web Services

Several mechanisms can be used for protecting messages exchanged through the Internet. They can be classified into two approaches: those that protect a message, or selected parts of a message, by creating an envelope that encapsulates the message; those that establish a secure connection on the network protocol stack.

WS-Security is an important work that follows the first approach [2]. It defines mechanisms that include integrity, confidentiality and authentication of SOAP messages. These mechanisms can be used independently or combined. WS-Security uses the W3C specification for "XML Signature" [3] and "XML Encryption" [8], and defines a way for embedding digital signatures and cryptographic information in SOAP message. FloCI-EE (Flexible Low-Cost Internet Extended-Enterprise) [7] is another work specifically conceived for protecting Web Services. This work provides authentication and authorization mechanisms for Web Services, combining role based access control (RBAC), public-key infrastructure (PKI) and privilege management infrastructure (PMI).

Other works that propose mechanisms for protecting XML content, but have not been specifically conceived for Web Services, can also be considered as potential solutions. For example, XS-Cube (XML Security Services Suite) defines a security system that allows implementing security mechanisms without modifying legacy systems [18]. The proposal consists in introducing a proxy server and a signature/verification server in the network. The proxy server is placed between the application and the Internet to examine the authenticity of XML messages exchanged between applications. Presently, the XS-Cube only implements message signature, but cryptography, key management and access control are planned enhancements.

Despite the existence of specific approaches for protecting XML content, traditional approaches such as SSL and IPsec can also be used with Web Services. The work described in this paper is based on the use of IPsec, but the proposal can be extended for handling other protection mechanisms, such as SSL or even approaches that protects XML content. IPsec was chosen because is a consolidated model, that provides mechanisms for authentication, confidentiality and integrity. Also, IPsec can be deployed without modifications in legacy applications. The policy based approach employed by IPsec is flexible, allowing developers selecting the most appropriated set of protections for specific scenarios. The disadvantage of IPsec, comparing with other approaches such as WS-Security, is the performance impact introduced by protecting the lower layers of the protocol stack: network and transport. However, this performance impact is compensated by the enhanced security provided by IPsec against attacks to the network infrastructure and host's operating system. Another disadvantage of IPsec is the need of reconfiguring firewalls, due to the modifications introduced in the lower network layers and the use of a key exchange protocol (IKE). In order to support IPsec, two sets of rules must be added. The first one is required for permitting the exchange of IKE packets (IKE uses the UDP protocol, port 500). The second set is required for permitting the traffic of IPsec packets. In this case, the firewall must allow the traffic of AH protocol (Authentication Header, protocol number 51), ESP (Encapsulating Security Payload, protocol number 50) and IPComp (Internet Protocol Compression, protocol number 108). In this work, it is assumed that the IPsec channel will be created between the server application (i.e., the Web Service) and the client application. Then, all firewalls between the client and server are assumed to be configured in order to support IPsec.

Considering the proposed solutions for protecting Web Services, WS-Security is probably the most important, because it is being supported by OASIS [15]. However, WS-Security doesn't intend to be a complete solution, instead, it intends to be part of a solution that integrates other security mechanisms embedded in the application or in network protocols. The work in this paper develops an issue not addressed by the WS-Security specification, which consists in including security policy requirements with the description of the services (i.e., the WSDL file).

3. Proposal of representation of IPsec policies on WSDL

This work proposes embedding IPsec policies description into WSDL documents. IPsec policies are described using a XML extension defined according to a formal schema. The XML model is based on the recent works of IETF regarding standardization of network policies. The IETF is defining a standardized model for representing policies on different areas of network management. The base for this model is the PCIM (Policy Core Information Model), defined by RFC 3060 [13]. PCIM has been complemented by RFC 3460, which was denominated PCIMe (Policy Core Information Model Extensions) [14]. The PCIM is an implementation independent object oriented information model. The model defines a standard way for representing network policies as aggregations of rules, described in terms of conditions e actions. PCIM e PCIMe are abstract models, and do not have sufficient elements for describing policies for particular areas of network administration. To address particular areas, PCIM needs to be extended. IETF has already defined some models based on PCIM extensions, including the IPsec policy model described by the RFC 3585 [10]. The IETF IPsec model is used in the proposal described in this paper. Fig. 1 illustrates the UML representation of its main classes. Many attributes of the classes were omitted in the Figure[1].

[1] The figure is based on the drafts published by IETF before the RFC 3585 publication.

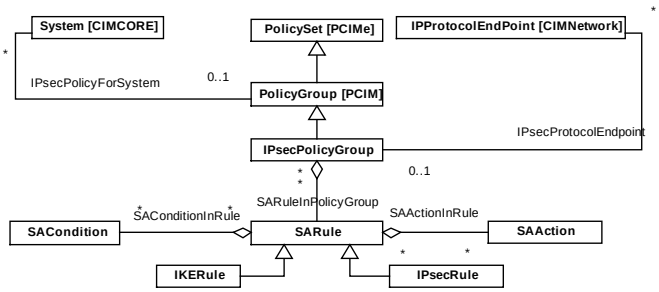

Figure 1: IETF IPsec policy model.

One observes in Fig. 1 that some classes are specialization of classes originally defined in PCIM and PICMe. The Figure also has references to the DMTF CIM model (Common Information Model) [6], used for describing network resources. The policies are represented by the class IPsecPolicyGroup, which can aggregate two types of rules: IKERule (for negotiations of the phase 1 of IKE) and IPsecRule (for negotiations of the phase 2 of IKE). The class IPsecPolicyGroup can aggregate instances of itself, through the aggregation class PolicySetComponent, inherited from PolicySet, allowing the construction of nested policies. The class IPProtocolEndpoint associated to class IPsecPolicyGroup represents the endpoint where the policy must be applied. Each policy contains an ordered group of rule instances. Each rule instance have conditions, represented by the class SACondition, and actions, represented by the class SAAction. A rule instance permits to map conditions to actions. The conditions represent traffic parameters (IP addresses, ports e protocols), and computer credentials (such as digital certificates or IKE identities[2]).

When some event triggers the policy evaluation of a policy (e.g., an inbound packet arriving in the network interface), the packet header information is used for evaluating the traffic conditions of the rules and to find the first compatible. The action of the first rule is then used to fill the parameters of the IKE negotiation (acceptable encryption algorithms, lifetime of SAs, etc.). Rules are evaluated sequentially, according to its policy group and priority. The action of a rule which conditions are satisfied is executed. The IPsec model ***hedges*** the use of SAActions to an ordered choice instead of a list of actions to be executed. Fig. 2 presents an example of IPsec policy using the IETF model. This example shows the policy for protecting the HTTP (TCP port 80) and SMTP (TCP port 25) traffic of a server. Any other traffic will be blocked. The policy is not completely described, and some elements and attributes have been omitted for simplifying the presentation. Also, IPsec rules must be created for inbound and outbound traffic. Only inbound traffic rules have been represented. The rules of the policy are grouped by an instance of the IPsecPolicyGroup class (named "SamplePolicy"). An instance of the IKERule class (named "Rule1") and an instance of the IPsecRule class (named "Rule2") where used to describe which traffic is allowed and the respective actions for protection. Another instance of the IKERule class (named "Rule3") was used to indicate that all traffic not matching the rules "Rule1" and "Rule2" must be blocked. Only the two first rules were completed represented in the Figure. The rule "Rule1" contains two conditions, which are filter conditions that represents the HTTP and SMTP traffic (the conditions are grouped using the OR operator, by using the attributes of the classes SARule and SAConditionInRule, omitted in the Figure). The rule "Rule2" has the same conditions than "Rule2" (the model supports reuse of condition).

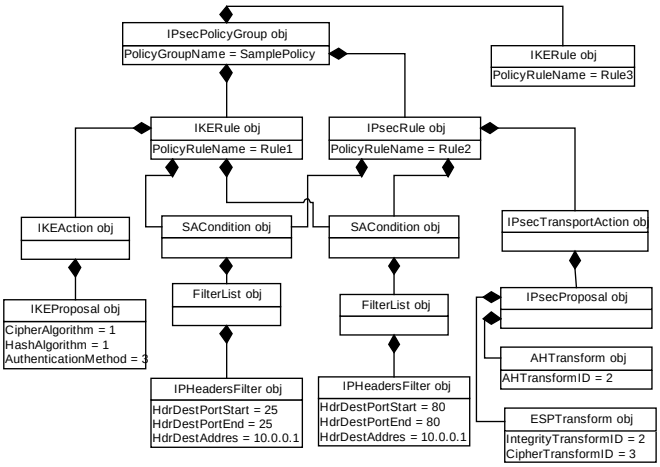

Figure 2. Example of IPsec policy using the IETF model.

If the conditions of the rule "Rule1" are satisfied, then the corresponding IKEAction action will be performed. Because the conditions of rule "Rule2" will be also satisfied, the IPsecAction action will also be performed. The instance of the IKEAction class has a "proposal", which consists of the following algorithms: DES-CBC for encryption, MD5 for hashing and RSA digital signature as authentication method. The IETF model represents protocols and algorithms using numbers instead of names (refer to IPSec DOI [16]). The instance of the IPsecAction class has a proposal that includes the protocols AH and ESP. MD5 was defined as the transformation algorithm for AH. SHA-1 and 3DES were defined, respectively, as integrity and encryption algorithms for ESP.

In order to enable embedding the IPsec policies descriptions into WSDL documents, this work proposes a XML schema for representing IPsec policies. There are

IETF directives instructing how to map the PCIM model on LDAP repositories, but nothing was published for XML mapping. As described in this section, the IETF IPsec policy model is composed by two types of classes: structural classes, representing the policy information, and associative classes, providing complementary information about the policies and indicating how the structural classes are related to each other. Two types of mapping are needed for representing the IETF information model into XML: Mapping of structural classes: it is basically a one-to-one mapping, where the classes represented in XML receive all attributes described in the specialized information model classes (abstract and generic classes are not represented). Mapping of associative classes: There are several approaches, but two has been used in this work: I) define XML elements for representing structural classes and mapping the attributes of the associative classes to these elements (the element that receives the attributes is defined according to the semantics of each association and its attributes) or II) define an XML element for representing the associative class.

Because there are innumerous possibilities for configuring IPsec, the whole XML schema for policy representation is quite extensive. This section will present only some aspects of schema, and will illustrate its utilization by examples. For a complete description of schema, please refer to [12].

```
<xsd:element name="SecurityPolicy">
   <xsd:complexType>
     <xsd:sequence>
       <xsd:element name="IPsecPolicyGroup"
                    type="IPsecPolicyGroupType"
                      maxOccurs="unbounded"/>
     </xsd:sequence>
     <xsd:attribute name="SecurityPolicyName" type="xsd:string"/>
     <xsd:attribute name="PolicyDecisionStrategy"
                    type="xsd:nonNegativeInteger"/>
   </xsd:complexType>
</xsd:element>
```

Figure 3. XML schema: IPsec policy root element.

```
<xsd:complexType name="IPsecPolicyGroupType">
   <xsd:sequence>
     <xsd:element name="IKERule"
                  type="IKERuleType" minOccurs="0"
                   maxOccurs="unbounded"/>
     <xsd:element name="IPsecRule"
                  type="IPsecRuleType" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xsd:element name="IPsecPolicyGroup"
                  type="IPsecPolicyGroupType"
                  minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="PolicyGroupName" type="xsd:string"/>
   <xsd:attribute name="PolicyDecisionStrategy"
                 type="xsd:nonNegativeInteger"/>
   <xsd:attribute name="PolicyRoles" type="xsd:string"/>
   <xsd:attribute name="Priority" type="xsd:nonNegativeInteger"/>
</xsd:complexType>
```

Figure 4. XML schema: policy group elements.

```
<definitions ...>
   ... Definitions of WSDL
   <SecurityPolicy Name="Sample">
     <IPsecPolicyGroup PolicyGroupName="SamplePolicy">
       <IKERule PolicyRuleName="Sample1">
         <SACondition PolicyConditionName="Condition1">
           <FilterList>
             <IPHeadersFilter HdrDestAddress="10.0.0.1"
               HdrDestPortStart="80" HdrDestPortEnd="80"/>
           </FilterList>
         </SACondition>
         <SACondition PolicyConditionName="Condition2">
           <FilterList>
             <IPHeadersFilter HdrDestAddress="10.0.0.1"
               HdrDestPortStart="25" HdrDestPortEnd="25"/>
           </FilterList>
         </SACondition>
         <IKEAction>
           <IKEProposal CipherAlgorithm="1" HashAlgorithm="1"
             AuthenticationMethod="3"/>
         </IKEAction>
       </IKERule>
       <IPsecRule PolicyRuleName="Rule2">
         <SACondition PolicyConditionName=" Condition1"/>
         <SACondition PolicyConditionName=" Condition2"/>
         <IPsecTransportAction>
           <IPsecProposal>
             <AHTransform AHTransformID="2" />
             <ESPTransform IntegrityTransformId="2" CipherTransformId="3"/>
           </IPsecProposal>
         </IPsecAction>
       </IPsecRule>
     </IPsecPolicyGroup>
   </SecurityPolicy>
</definitions>
```

Figure 5: Example of IPsec policy using the proposed XML model.

In order to illustrate the approach used for mapping the IETF model into XML, some parts of the proposed XML schema are shown in Figures 3 and 4. The root element of the IPsec security policy is shown in Fig. 3. The root element SecurityPolicy has two attributes: SecurityPolicyName, which defines the security policy name and PolicyDecisionStrategy, which defines the strategy for evaluating the policy. The schema for representing IPsecPolicyGroup is shown in the Fig. 4. IPsecPolicyGroup represents the class of same name, defined by IETF (see in Fig. 2). As well as in the IETF model, an IPsec policy group contains rules for describing negotiation of phase 1 (IKERule) and phase 2 (IPsecRule), and can also contain other IPsec policy groups.

Fig. 5 presents an example of security policy using the proposed model. The policy represented in XML is the same represented using the IETF information model in Fig. 2. The PolicyGroup named “SamplePolicy” contains two rules, representing both phases of IPsec negotiation. This example also illustrates the use of reusable conditions. The first rule (“Rule1”) defines two conditions, for inbound HTTP and SMTP traffic. The second rule reuses the conditions of the first rule, because it uses the same conditions names. The device will create an IPsec channel for the traffic that satisfies those conditions. In the first phase, DES-CBC protocol is used

for encryption, MD5 for hashing and RSA digital signatures for authentication (the protocols and algorithms are represented by numbers, defined in [16]). In the second phase, MD5 is used for AH hashing, SHA-1 for ESP integrity and 3DES for ESP encryption.

```
<?xml version="1.0"?>
<definitions name="StockQuote"  ...>
 <types> ... </types>
 <message name="..."> ... </message>
 <portType name="...">
  <operation name="...">
  </operation>
 </portType>
 <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
 </binding>
   <service name="StockQuoteSevice">
     <documentation>My First Service</documentation>
     <port name="StockQuotePort" binding="tns:StockQuoteBinding"
                  securityPolicy="StockQuoteSecurityPolicy">
          <SOAP:Address location="http://www.example/"/>
     </port>
   </service>
   <SecurityPolicy Name="StockQuoteSecurityPolicy">
    ...
   </SecurityPolicy>
</definition>
```

Figure 6. WSDL with embedded security policies.

The WSDL specification supports language extensions without modifying its specifications [5]. Even data types and protocols can be extended. Fig. 6 illustrates how to embed the security policy description into a WSDL document. An IPsec policy described in XML is referred in a WSDL document by an attribute called SecurityPolicy, appended in the element *port* of the session *service*. This attribute indicates the name of security policy. The description of the security policy, represented by the element <SecurityPolicy>, is inserted as a child element of the element *description*.

4. Framework for embedding IPsec in Web Services

This section describes a framework used to automate the process of interpreting and applying IPsec policies described using the proposed XML model. The framework defines an API for interpreting and applying the IPsec policies received from the Web Service on the client's (i.e. the Web Service consumer) operating system. The API help developers by offering a systematic approach for embedding IPsec negotiations in Web Service consumer applications, without requiring advanced knowledge on IPsec configuration or XML file manipulation. The consumer application, using the available methods, can obtain, apply and remove the IPsec policies on the operating system without directly manipulating the WSDL files or IPsec commands on the underlying operating system. The API defines a SecurityManager class with 6 methods, as described in Table 1:

Table 1. API for manipulation of IPsec policies

Method	Description
SecurityManager	Constructor. Input parameters: i) the XSL file for transforming IPsec policies into IPsec commands for the underlying IPsec implementation. ii) the XSL file that is used to extract the policies that satisfy the client's security requirements (described further, in this section)
loadPolicyFile loadPolicyFromURL	Loads the IPsec policy embedded in the WSDL document and extract from the policies proposed by the Web Service those that satisfy the client's security requirements. Input parameter: the URL or directory path of the WSDL document.
applyPolicy	Configure the IPsec policy in the underlying system (operating system's or other IPsec implementation).
removePolicy	Remove the IPsec policy from the underlying system.
verifyIPsecChannel	Checks if the IPsec channel has been correctly built. This method can be called only after calling the Web Service. In case of a Web Service call failure, this method permit to determine if the cause was the IPsec configuration.

That API was developed in Java. The source code is described on [12], and the code is available for download on [11]. The API uses XSL (Extensible Stylesheet Language) [1] transformations for transforming the XML IPsec policies descriptions into IPsec configuration commands to the underlying operating system. By creating the appropriated XSL transformation document, the API has can be easily ported to different platforms and IPsec implementations. An excerpt of a XSL file for transforming policies into commands for applying policies to Windows 2000 hosts is shown in section 5. Another XSL document is used for creating the commands for removing the policies.

The Fig. 7 presents the interaction between the API and the other parts acting in a Web Service call. After obtaining the document that describes the security policies for accessing a particular Web Service, the Web Service consumer (i.e., a client) can check if those policies attend the minimal requirements. The file of minimal requirements contains the policies allowed by the client for creating IPsec channels. This file can contain many policies, which will appear in order of preference. The Web Service can offer a range of policies to access its service, but the API will extract only one policy that satisfies its requirements, based on the policies priorities. It is the Web Service consumer administrator the responsibility of defining the policy priorities, based on security, performance or other criteria. By selecting only the policy with the highest priority, one avoids the need of re-negotiating IPsec policies when creating the IPsec channel. The minimal requirements can be described in terms of protocols, algorithms, keys size, acceptable certification authorities or other parameters desired by the client.

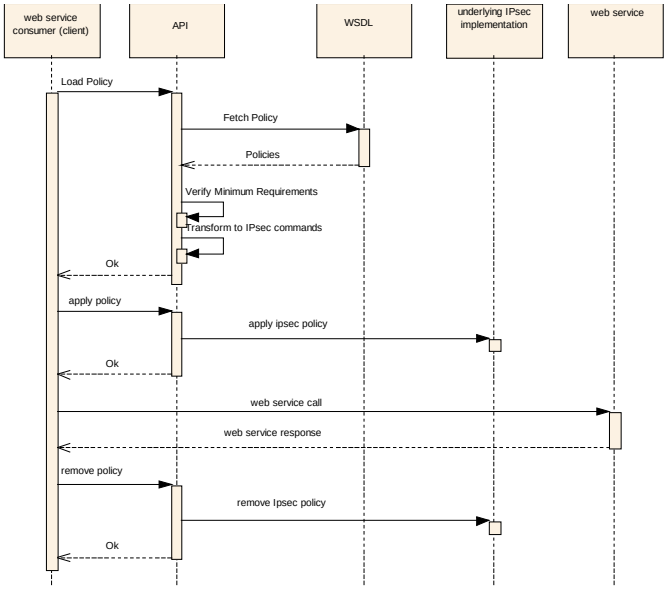

Figure 7. UML Sequence diagram and API calls

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/xhtml1/strict">
    ...
    <xsl:template match="IKEAction">
      <xsl:text>
          </xsl:text>
      <xsl:copy>
      <xsl:goes-each select="@*">
        <xsl:copy/>
      </xsl:goes-each>
      <xsl:goes-each select="*[self::IKEProposal]">
        <xsl:if test="@CipherAlgorithm='5'">
          <xsl:text>
            </xsl:text>
          <xsl:copy-of select="."/>
        </xsl:if>
      </xsl:goes-each>
      <xsl:text>
          </xsl:text>
      </xsl:copy>
    </xsl:template>
    <xsl:template match="IPsecTransportAction">
      <xsl:text>
          </xsl:text>
      <xsl:copy>
      <xsl:goes-each select="@*">
        <xsl:copy/>
      </xsl:goes-each>
      <xsl:goes-each select="*">
        <xsl:if test="AHTransform/@AHTransformID='3' and
          ESPTransform/@CipherTransformId='3'">
          <xsl:text>
            </xsl:text>
          <xsl:copy-of select="."/>
        </xsl:if>
      </xsl:goes-each>
      <xsl:text>
          </xsl:text>
      </xsl:copy>
    </xsl:template>
    ...
</xsl:stylesheet>
```

Figure 8. Excerpt of a document of minimal requirements

The client's minimum-security requirements are represented by a XSL document [1]. Fig. 8 presents an excerpt of a sample minimum requirement document. The complete XSL document can be found in [12]. The XSL document transform the IPsec document received form the Web Service into another IPsec document that contains all rules and conditions, but only the desired actions. The requirement filters specified in the XSL document are: (highlighted in the Fig. 8): Use the encryption 3DES algorithm (represented by the numeral 5) in the first phase of IKE; Use the SHA integrity algorithm (represented by the numeral 3) for the AH (transport or tunnel mode); Use the 3DES encryption algorithm (represented by the numeral 3) for the ESP protocol (transport or tunnel mode).

5. Implementation

The policy model and the framework developed in this work are platform independent. The policy model, based on IETF standard, permits to represent several policy features so far not available in most current IPsec implementations. However, the "transformation approach" adopted in this paper permits to easily adapt the XML IPsec policy information to specific platform implementations. This section will present the "transformations" required for supporting the Microsoft Windows 2000 IPsec implementation. Windows 2000 has been chosen because it offers a flexible implementation for IPsec policies, permitting to validate most elements in the IPsec policy model described in this work. Other IPsec implementations, such as FreeS/WAN (www.freeswan.org) offers comparatively less options for describing IPsec policies.

The native IPsec implementation in Windows 2000 offers a command line utility (named "*ipsecpol*") that permits to create, apply and remove policies from the operating system. The policies embedded in the WSDL files are transformed into Windows 2000 IPsec commands, using a set of API's that manipulates XSL documents, as described in section 4. The proposed framework was easily adapted for Windows 2000 by writing a XSL document for transforming the IPsec policies embedded in the WSDL files (see Fig. 9) into *ipsecpol* commands. Fig. 10 shows a short excerpt of the XSL document for extracting the *PolicyGroup* name from a WSDL file and creating a IPsec policy. The XSL output is a executable ".bat" file.

The templates "IKERule" and "IPsecRule" will transform, respectively, IPsec Phase 1 rules and IPsec Phase 2 rules into *ipsecpol* commands. The IPsecRule template is shown in Fig. 11. In this case, the *ipsecpol* command includes the flags –f, which defines the filtering conditions for the rule, and –n, which defines the action

for the rule. The actions corresponding to the rule are defined in terms of other templates that receives the parameters of the action and rule to be transformed. These templates are not shown in this paper, but its structure is similar to the XSL code excerpts presented in Fig. 10 and 11.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://com.test/wsdl/MyPing"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    targetNamespace="http://com.test/wsdl/MyPing" name="MyPing">
   <types />
   <message name="PingIF_sayPing">...</message>
   <message name="PingIF_sayPingResponse">...</message>
   <portType name="PingIF">...</portType>
   <binding name="PingIFBinding" type="tns:PingIF">...</binding>
   <service name="MyPing">...</service>
   <SecurityPolicy SecurityPolicyName="StockQuoteSecurity">
    <IPsecPolicyGroup PolicyGroupName="Politica1">
     <IKERule PolicyRuleName="IPsecRule">
       <SACondition PolicyConditionName="Condicao1">
         <FilterList Direction="4">
           <IPHeadersFilter
              HdrSrcAddress="192.168.200.198" HdrSrcMask="255.255.255.255"
              HdrDestAddress="192.168.200.0" HdrDestMask="255.255.255.0"
              HdrDestPortStart="" />
         </FilterList>
       </SACondition>
       <IKEAction>
         <IKEProposal CipherAlgorithm="1" HashAlgorithm="1"
    AuthenticationMethod="PRESHARE:abc" MaxLifetimeSeconds=""
    MaxLifetimeKilobytes="" GroupId="1" />
       </IKEAction>
     </IKERule>
     <IPsecRule PolicyRuleName="IPsecRule">
       <SACondition PolicyConditionName="Condicao1" />
       <IPsecTransportAction>
         <IPsecProposal>
           <AHTransform AHTransformID="2" />
           <ESPTransform CipherTransformId="2" />
         </IPsecProposal>
       </IPsecTransportAction>
     </IPsecRule>
    </IPsecPolicyGroup>
   </SecurityPolicy>
```

Fig. 9. Policy Embedded in a WSDL File

```
<xsl:template match="IPsecPolicyGroup">
echo Policy: <xsl:value-of select="@PolicyGroupName"/>
<xsl:apply-templates select="IKERule"/>
<xsl:apply-templates select="IPsecRule"/>
<xsl:text>
ipsecpol</xsl:text>
<xsl:text> -w REG -p "</xsl:text>
<xsl:value-of select="@PolicyGroupName"/><xsl:text>" -x </xsl:text>
</xsl:template>
```

Figure 10. IPsec commands: Policy Creation

Note that by defining an XSL file for transforming the policy into commands, the API described in the Table 1 (section 4) needs to suffer only small adaptations in order to support other operating systems and platforms.

For the evaluation tests, a Web Service that implements an "echo" method was chosen as a "sample" application. The "echo" method was used to measure the round trip delay in Web Services calls. The Web Service receives a string of characters and returns the same string to the Web Service consumer (similar to the IP "ping"). The Web Service was evaluated with several combinations of protocols and algorithms for encryption and hashing, as illustrated in Table 2. Each combination have been represented as a distinct policy and embedded into the WSDL files using the proposed IPsec XML scheme. Figure 9 shows the WSDF file (some parts have been omitted) with the IPsec policies embedded for one scenario in Table 2.

```
<xsl:template match="IPsecRule">
  echo Rule: <xsl:value-of select="@Name"/>
  <xsl:text> ipsecpol</xsl:text>
  <xsl:text> -w REG -p "</xsl:text>
  <xsl:value-of select="../@PolicyGroupName"/><xsl:text>" -r "</xsl:text>
  <xsl:value-of select="@PolicyRuleName"/><xsl:text>" </xsl:text>
  <xsl:text> -f</xsl:text>
  <xsl:apply-templates select="SACondition"/>
     <xsl:text> -n </xsl:text>
     <xsl:for-each select="IPsecTransportAction">
          <xsl:apply-templates select="."/>
          <xsl:if test="@PolicyActionName">
               <xsl:variable name="ActionName" select="@PolicyActionName"/>
               <xsl:variable name="Rule" select=".."/>
               <xsl:apply-templates select="//IPsecTransportAction
               [attribute::PolicyActionName=$ActionName and ..!=$Rule]"/>
          </xsl:if>
     </xsl:for-each>
     <xsl:for-each select="IPsecTunnelAction">
          <xsl:apply-templates select="."/>
          <xsl:if test="@PolicyActionName">
               <xsl:variable name="ActionName" select="@PolicyActionName"/>
               <xsl:variable name="Rule" select=".."/>
               <xsl:apply-templates select="//IPsecTunnelAction
                    [attribute::PolicyActionName=$ActionName and ..!=$Rule]"/>
          </xsl:if>
     </xsl:for-each>
     </xsl:template>
```

Figure 11. IPsec commands: Adding a IPsec Rule

The tests were implemented using two computers: a Pentium III 900MHz as client and an AMD Duron 850MHz as server, both with 192MB of RAM. It was used a single cable (cross-over) to interconnect both computers, simulating a 100 Mbps network. Both equipments have installed the Microsoft Windows 2000 Professional operating system. The Apache Tomcat version 4.1 server was used for hosting the Web Service. The Web Service was developed using the JSWDP 1.1 (Java Web Services Developer Pack) and the Java 2 Standard Edition 1.4.1, both from Sun Microsystem.

Table 2. Round-trip delay

	1ª call			Other calls		
	Avg (ms)	Dev	Imp. (%)	Avg (ms)	Dev.	Imp. (%)
Without IPsec	424,00	4,67	-	9,37	2,49	-
AH[MD5]	493,89	7,01	16,48	9,88	2,42	5,40
ESP[MD5]	497,44	16,69	17,32	9,89	2,77	5,48
AH[SHA]	500,78	8,38	18,11	9,93	2,40	5,93
ESP[SHA]	500,56	8,68	18,06	9,98	2,43	6,43
ESP[DES]	499,67	15,54	17,85	10,48	2,70	11,80
AH[MD5]+ ESP[DES]	497,56	4,93	17,35	10,78	2,51	15,03
ESP[3DES]	487,44	38,35	14,96	11,41	3,60	21,71

Table 2 describes the average round trip delays of Web Services "echo calls" with messages of 1000 bytes. Once

the IPsec was established, the echo call was repeated 1000 times. The time spent in the first call was separately registered, because it includes the time spent for creating the IPsec channel (including the interpretation of the WSDL file and the embedded IPsec policies). In order to obtain the variation of negotiation time (first call), the application was executed 10 times for each protocol combination. The "ping" message size is configurable. The tests have been carried out using messages 10, 1.000 e 10.000 bytes. Besides the "ping" message, the IP packets generated in the Web Service call include the SOAP and HTTP headers, as well as the headers of lower layer protocols (IP, TCP, Ethernet) and IPsec (when used). The tests shown that the use of IPsec can increment the packets size up to 59 bytes (24 bytes for AH with MD5, from 19 to 23 bytes for ESP with DES, from 23 to 27 bytes for ESP with MD5 and of form 31 to 35 bytes for ESP with DES and MD5. The maximum overhead was achieved when AH was used with ESP).

Table 2 shows the results of the evaluation considering messages of 1000 bytes. The table is ordered by the average time of calls not including the first call (the one that includes the IPsec negotiation). The "impact" represents the percentage increase of the round trip delay when comparing the call with and without IPsec. The impact results for message sizes of 10 and 10000 were very similar to those shown in Table 2.

6. Conclusion

The evaluation tests shown that, by properly selecting the strategy for including the IPsec policy description in the WSDL file, the modifications does not interfere with existing tools for developing Web Services applications. The evaluation tests also shown that the proposed model was able to correctly represent IPsec policies, for diverse configurations. Future works will be the extension of the security policy model for representing other protecting mechanisms, such as SSL and WS-Security. The framework also requires the development of a tool for automating the process of building the XSL documents that describes the client's minimum requirements. Also, the development of a mechanism for selecting policies on the server side based on the client's identifications would be a significant enhancement to this work.

7. References

[1] Adler, S., Berglund, A., Caruso, J., Deach, S., Graham, T., Grosso, P., Gutentag, E., Milowski, A., Parnell, S., Richman, J., Zilles, S. *Extensible Stylesheet Language (XSL) Versão 1.0*. W3C Recommendation, http://www.w3.org/TR/xsl, October 2001.

[2] Atkinson, B., Della-Libera, G., Hada, S., Hondo, M., Hallam-Baker, P., Klein, J., LaMacchia, B., Leach, P., Manferdelli, J., Maruyama, H., Nadalin, A., Nagaratnam, N., Prafullchandra, H., Shewchuk, J., Simon, D. *Web Services Security (WS-Security) Version 1.0*, http://www-106.ibm.com/developerworks/webservices/library/ws-secure/. April 2002.

[3] Bartel, M., Boyer, J., Fox, B., LaMacchia, B., Simon, E. *XML-Signature Syntax and Processing*, W3C Proposed Recommendation, http://www.w3.org/TR/2001/PR-xmldsig-core-20010820/, August 2001.

[4] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D. *Simple Object Access Protocol (SOAP) 1.1*. W3C Note, http://www.w3.org/TR/SOAP/, May 2000.

[5] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S. *Web Services Description Language (WSDL) 1.1*. W3C Note, http://www.w3.org/TR/wsdl, March 2001.

[6] Westerinen, A., Strassner, J. *CIM Core Model White Paper: Common Information Model (CIM) Core Model, Version 2.4.* DSP111. Distributed Management Task Force (DMTF), August 2000.

[7] Fürst, K., Schmidt, T. Wippel, G. *Flexible Low-Cost Internet Extended-Enterprise (FloCI-EE)*. IEEE Internet Computing, October 2002.

[8] Imamura, T., Dillaway, B., Simon , E. *XML Encryption Syntax and Processing*. W3C Recommendation, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, December 2002.

[9] IETF IPsec Charter, http://www.ietf.org/html.charters/ipsec-charter.html.

[10] Jason, J., Rafalow, L., Vyncke, E. *IPsec Configuration Policy Model* RFC 3585, August 2003.

[11] Jamhour, E., Mecanismo de Segurança para Web Services Baseado em IPsec, http://www.ppgia.pucpr.br/~jamhour/WSIPsec, August 2003.

[12] Kropiwiec, C.D., Proposta de um Mecanismo de Segurança para Web Services Baseado em IPsec, Dissertação do Metrado em Informática Aplicada, PUCPR, 193 p., August 2003.

[13] Moore, B. e E. Elleson, J. Strassner. *Policy Core Information Model – Version 1 Specification*, RFC 3060, February 2001.

[14] Moore, B. Policy Core Information Model (PCIM) Extensions. Internet RFC 3460, January 2003.

[15] Organization for the Advancement of Structured Information Standards – OASIS – OASIS Web Services Security TC http://www.oasis-open.org.

[16] Piper, D. *The Internet IP Security Domain of Interpretation for ISAKMP*. Internet RFC 2407. November 1998.

[17] Roy, J.; Ramanujan, A. *Understanding Web* services. IT Professional , Vol: 3 Issue: 6 , Nov/Dec 2001, p: 69 -73.

[18] Takase, T.; Uramoto, N.; Baba, K. *XML digital signature system independent of existing applications*. Applications and the Internet (SAINT) Workshops, 2002. Proceedings, 2002, Symposium on , 2002. p: 150 -157.