Optimizing Web Services Security Processing Models

I.E. Suriarachchi and N.S. Mihindukulasooriya
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.

Abstract — **Security is a major concern of today's enterprise web services due to its message oriented nature. Web services messages containing confidential information can be transmitted on unsecured networks thus should have proper mechanisms to protect them possible attacks. To cater those requirements, Web Services Security specification defines enhancements to SOAP messaging providing authentication, message integrity and confidentiality without losing the interoperability. Security requirements and capabilities of web services are expressed using Security Policy language. Thus security policy processing plays a vital role in any web service security engine. Security processing model should be efficient and invincible to possible attacks.**

In this paper, we evaluate the current web service security processing models and discuss their weaknesses. We propose an improved security processing model for web services security which is more efficient and less vulnerable to attacks such as denial of service attacks.

Index Terms — **Denial of Service Attacks, Performance Optimization, Security, Security Policy, Web services**

I. Introduction

With the wide adoption of service oriented architecture in the industry, web services have influenced of most of the software systems. Web services mostly use SOAP protocol [2] for message exchange. SOAP message construct consists of a SOAP envelope which has a mandatory SOAP body and an optional SOAP header. SOAP body is used for actual message payload while SOAP header carries the Meta data about the message. Web Services Security specification [1] defines how security related information should be conveyed using a security header block in the SOAP header. Web Services Security specification defines a standard way of constructing and validating SOAP security header so that it will interoperable in all the web services stacks.

Web services security policy specification [4] which is an extension of web services policy framework [3] defines a standard way of expressing security requirements and capabilities of a web service in an interoperable manner.

```
<soapenv:Envelope>
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1">
      <wsse:UsernameToken>
        <wsse:Username>alice</wsse:Username>
        <wsse:Password Type="..#PasswordText">
          bobPW
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <ns:add>
      <param0>2</param0>
      <param1>6</param2>
    </ns:add>
  </soapenv:Body>
</soapenv:Envelope>
```

Fig. 1. A secure SOAP message including the mandatory SOAP body element and a SOAP header with a security header block. Security header contains a Username token which used to provide authentication in web services security.

The web services policy framework provides a general purpose model and corresponding syntax to describe the policies of a web service. Web services policy framework which is built on web services policy specification defines a base set of constructs that can be used and extended by other domain specific web services policy specifications to describe a broad range of service requirements and capabilities. A policy, according the specification is a collection of policy alternatives and a policy alternative is a collection of policy assertions where a policy assertion represents an individual requirement, capability, or other property of a behavior of a web service. Web services security policy specification defines a set of standard policy assertions to express security requirements and constraints.

```
<wsp:Policy wsu:Id="UsernameTokenPolicy">
    <wsp:ExactlyOne>
      <wsp:All>
        <sp:SupportingTokens>
          <wsp:Policy>
            <sp:UsernameToken/>
          </wsp:Policy>
        </sp:SupportingTokens>
      </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 2. A security policy sample with a single policy alternative. This policy express the requirement that the request should accompany a Username Token. These assertions are defined in WS – Security Policy specification.

II. Web Services Security Processing Models

Web service security policy model represents the semantics of constructing a secure web service message according to the corresponding policy and also validating a secure message according to its applicable policy.

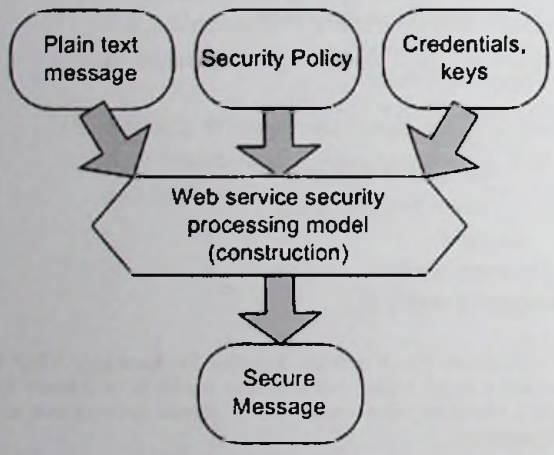

Fig. 3. Building a secure message. Secure message construction logic performs the cryptographic operations on the plain text message and generate secure message with security meta data included in the security header according to the security policy. Credentials, keys are used for cryptographic operations.

Building a secure message consists of two major tasks including performing cryptographic operations on the plain text message to fulfill the requirements expressed in the security policy and constructing a security header with the meta data required for the validation of the secure message according web services security specification.

Validation of a secure message also consists of two major tasks including doing a protocol validation of cryptographic and other security mechanisms used and also doing a policy validation to make sure all the security requirements and constraints expressed in the policy are met. In protocol validation task, elements in the security header of the SOAP message is validated according to the web services security, XML signature and XML encryption specifications. This includes username token validation with user information for authentication, time-stamp validation to prevent replay attacks, signature verification for integrity and non-repudiation, decryption for confidentiality and etc. In the policy validation task, the message is checked for all the security requirements expressed in the policy. For example, a policy can express which parts of the SOAP message must be signed or encrypted. Let's say there is a policy which specifies that body of the message and addressing header must be signed. A client can send a message only singing the body of the message with a valid signature. Even though this valid signature web services security processing model should reject this message because the addressing header is not signed. Thus, the objective of the policy validation task is to make sure that all the security requirements mentioned in the policy are satisfied.

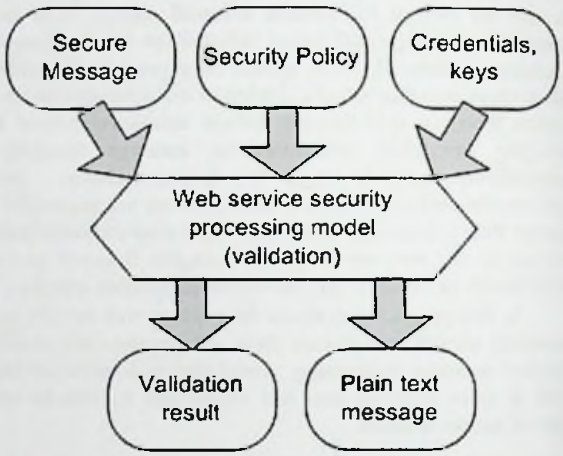

Fig. 4. Validating a secure message. Secure message validation logics performs cryptographic operations according to the meta data in the security header of the secure message and validates the message against the corresponding security policy. Credentials, keys are used for validation.

There are two main web service processing model classes based on how they perform security policy validation, namely two pass security processing model and integrated security processing model.

III. Two Pass Security Processing Model

In the two pass security processing model, security protocol validation and security policy validation is done in two phases. In the first phase, all the elements in the security header are validated not taking the policy in to account. This phase also extract information about what was present in the security header and how the message has been secured. Then these data is fed in to second phase, policy validation. In the policy validation phase, policy validation algorithms run on the extracted data to verify all security requirements and constraints enforced by the policy are satisfied.

One of the advantages of this model is it allows policy validation algorithms to evolve without much effect. For example, this model can be extended to process multiple policy alternatives without increasing the complexity to a

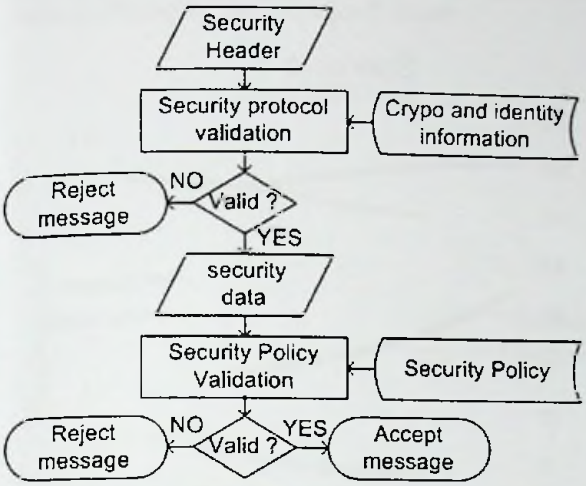

Fig. 5. Two pass security processing model. In this model, validation is done at two phases. In the first place security protocol validation is done and security policy validation is done at the second phase.

greater deal. At the same time, separation of policy validation to a second phase leads to main disadvantage of this model. The main disadvantage of this model is policy violations are found only in the second phase. This can make this model vulnerable to attacks like denial of service attacks.

IV. Denial of Service (DoS) Attacks

Major goals of information security are to protect the confidentiality, integrity and availability of information. A denial of service attack is an attempt to make a service or a resource unavailable to its legitimate users. Most common method of DoS attack is to saturate the service with large number of requests which will lead to resource exhaustion at the service provider. This will eventually lead the service provider to crash or unable to process further legitimate requests. Thus this is an attack on service availability.

Now let's see how the two pass security processing model is vulnerable to DoS attacks. An attacker can send large number of encrypted message to the service without worrying about the other requirements of the security policy. On public key infrastructure attacker only needs to know the public key of the service which is publically available in order to generate an encrypted message for the service. Thus the encrypted messages are valid according to XML encryption specification and will pass the first phase. Security policy validation will only be detected in the policy validation phase and messages will be rejected in second phase. Decryption is a compute-intensive cryptographic operation and having to decrypt large number of messages unnecessarily is wastage of resources. Even though this model prevents the service being exposed to attackers this can lead to resource exhaustion at the service provider. So this is a threat against the availability of the service.

V. Integrated Security Processing Model

Integrated security processing model tries to overcome the issues in two pass security processing model and provide better security policy validation model. The main objectives of integrated security processing model are

1) Preventing Denial of Service (DoS) attacks
2) Improving the efficiency of security policy processing

In the integrated security processing model, both security protocol validation and policy validation is done simultaneously. In this model, security protocol validation is made policy aware. Thus policy violations are detected before further processing preventing the risk of denial of service attacks. Integrated security processing model also eliminates the need to run two iterations to do the complete validation as both protocol validation and policy validation done in a single iteration. This removes the requirement of populating and managing intermediate data about security header at the protocol validation layer which will also contribute to the efficiency of the new security processing model.

This model can be further improved by adding a pre policy validation before the actual policy aware protocol validation phase. Here we use the principle of rejecting the invalid messages as early as possible without wasting computational resources for further processing. Structure of the security header can be deduced from the applicable policy of the web service and the derived structure can be used to do a quick validation by traversing only through the top level elements of the security header without doing any processing. This enables the security processing engine to reject the invalid messages as earliest possible stage without doing any further processing.

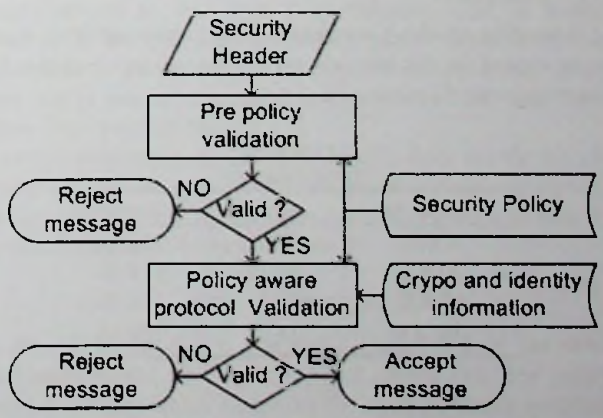

Fig. 6. Integrated security processing model with pre – policy validation.

VI. Apache Rampart and Rampart2

Apache Axis2 is an open source web services engine which is extensively used by the industry. Apache Rampart is the security module of Axis2 which extends the Axis2 engine with Web Services Security functionality. Apache Rampart

supports Web Services Security, Web Services Secure Conversation, Web Services Trust and Web Services Security Policy specifications. Apache Rampart engine uses a two pass security processing model. Thus it is used to measure performance of two pass security processing model. Ramaprt2, the next generation of Apache Rampart implements an integrated security processing model. Rampart2 is used to measure performance of integrated security processing model.

VII. Results

Apache Rampart and Rampart modules were used to measure the performance of each of these security processing models. This section presents the results of three scenarios that were tested with both implementations. Requests/sec was measured by varying the payload size.

Same asymmetric binding security policy was used for all three scenarios. Here are the requirements and constraints described in the security policy.

TABLE I
SECURITY REQUIREMENTS AND CONSTRAINTS

Property	Value
Initiator token	X509 certificate
Recipient Token	X509 certificate
Algorithm suite	Basic256
Singed Parts	Message body
	Binary security tokens
Encrypted Parts	Message body
	Message Signature
Protection order	Encrypt before sign

Complete security policy used is available in Appendix A.

A. Scenario 01

According to above mentioned policy, message body must be encrypted. In this scenario, an invalid message was sent by not encrypting the message body.

TABLE II
TEST SCENARIO 01

Payload Size	Rampart	Rampart2
1 KB	71.62	122.38
30 KB	52.12	119.44
60 KB	41.53	114.93
100 KB	34.78	109.33

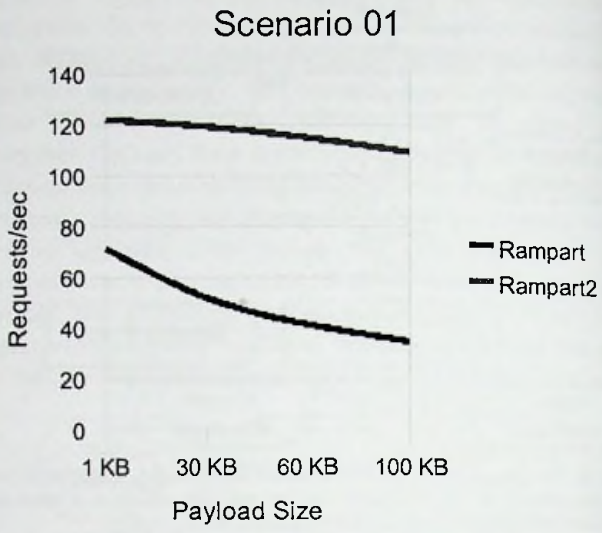

Fig. 7. Perfonnance comparison of Apache Rampart and Rampart for the scenario 01.

Above table and graph shows the results which are obtained in scenario 01. This clearly shows that Requests/sec value for Rampart2 doesn't depend much on the payload size. But it reduces for Rampart when the payload size is increasing. The reason for this is the optimized message processing model in Rampart2. In the security protocol validation stage of Rampart, it performs all validation steps and doesn't identify that the body is not encrypted until it comes into security policy validation stage. Therefore, processing time depends on the payload size. But Rampart2 identifies the policy violation very early as it's protocol validation is policy aware. Therefore, it doesn't process the message and processing time does not depend on the payload size.

B. Scenario 02

According to our policy, message parts should be encrypted before signing. In this scenario, an invalid message was sent by signing the message before encrypting. This is a violation of the protection order.

TABLE III
TEST SCENARIO 02

Payload Size	Rampart	Rampart2
1 KB	24.26	124.16
30 KB	15.95	121.55
60 KB	10.60	115.27
100 KB	07.35	110.69

In this scenario also, Rampart2 shows similar results to scenario 01. Rampart2's processing model identifies this policy

violation before going into protocol validations and rejects the message. Therefore, again Rampart2 doesn't

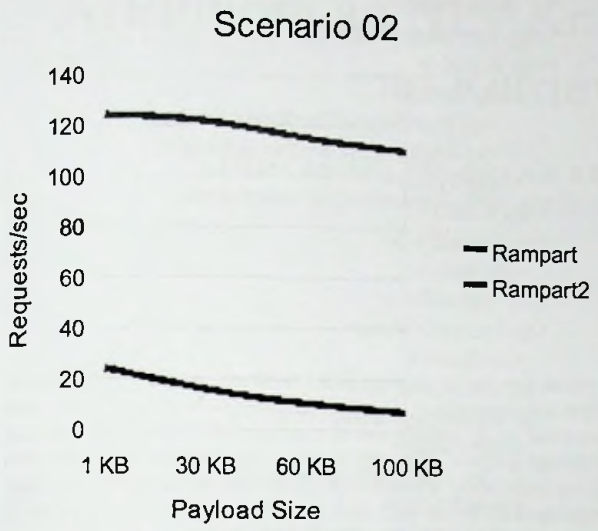

Fig. 8. Performance comparison of Apache Rampart and Rampart for the scenario 02.

perform any heavy operations. For Rampart, this scenario shows the same pattern of reducing the Requests/sec value with increasing payload size. That is because it again validates protocols before identifying policy violation in the second phase of the message processing. But the Requests/sec values are lower for Rampart in this scenario compared to scenario 01. That is because the body is encrypted this time and Rampart performs all cryptographic operations before policy validation.

C. *Scenario 03*

Finally in this scenario, a valid message according to the above policy was sent.

TABLE IV
TEST SCENARIO 03

Payload Size	Rampart	Rampart2
1 KB	22.13	34.39
30 KB	14.90	23.94
60 KB	10.08	16.87
100 KB	07.27	13.60

Rampart2 is considerably faster than Rampart in this valid message scenario as well. A message processing model can hardly affect the results of a valid scenario. Because all protocol validations and policy validations should be done anyhow. But as Rampart's processing model is two phase and Rampar2's one is single phase, there can be an improvement in Rampart2's performances. Rampart constructs additional data structures to hold results of protocol validations and validate those structures with security policy. But Rampart2 performs both validations in parallel.

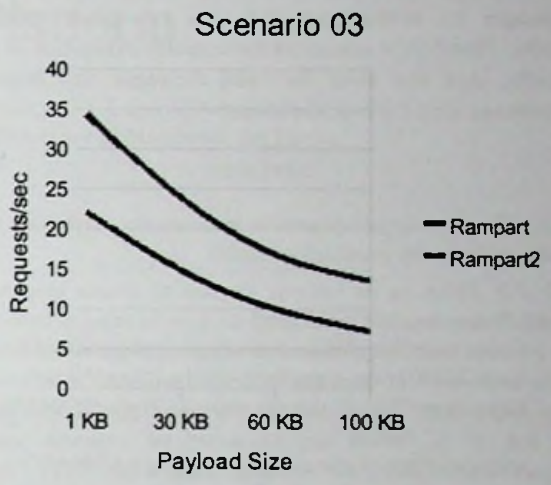

Fig. 9. Performance comparison of Apache Rampart and Rampart for the scenario 03.

In addition to that, Rampart2 uses Axiom as the xml info-set representation model throughout. But the bottom layer of Rampart (Apache WSS4J) uses DOM and a conversion between the two models (called DOOM) is done in a middle layer. This also can support the improved performances in Rampart2.

The point which is addressed in this paper is the contribution of an improved processing model for better performances. In scenario 01 and scenario 02, there are huge improvements in Rampart2 over Rampart. That is mostly because of the improved processing model which is used in Rampart2. In scenario 03 also Rampart2 showed better values and that is due to processing model improvement and some other improvements as well.

Security processing model of Rampart2, does not do the pre policy validation step of the proposed security processing model yet. So these results show the improvements of security protocol validation being policy aware.

VIII. Future Work

As mentioned above, Rampart2 doesn't follow the exact validation model which is described above. The "Pre policy validation" step which compares the message header structure with the security policy is still to be implemented in Rampart2. It will further improve the performance results of Rampart2.

Another area that needs further research is handling alternative policies in the integrated security processing model. Handling alternative polices becomes a tedious task when policy validation is done simultaneously with security protocol validation.

IX. Conclusion

Above test results clearly show that Rampar2 with policy aware protocol processing, shows an extremely high Requests/sec value for invalid messages compared to Rampart. This implies that this processing model rejects invalid messages for earlier compared to a two phase validation model. Therefore it minimizes the resource exhaustion in DOS attacks. And also even for valid messages, this improved processing model is much efficient.

Appendix

A. The security policy used to measure the performance of the two security processing models.

```
<wsp:Policy wsu:Id="perfTests"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
    oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/
                policy"
    xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/
                securitypolicy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:AsymmetricBinding
        <wsp:Policy>
          <sp:InitiatorToken>
            <wsp:Policy>
              <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/sec
urity policy/IncludeToken/AlwaysToRecipient">
                <wsp:Policy>
                  <sp:WssX509V3Token10/>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:InitiatorToken>
          <sp:RecipientToken>
            <wsp:Policy>
              <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/sec
urity policy/IncludeToken/Never">
                <wsp:Policy>
                  <sp:WssX509V3Token10/>
                </wsp:Policy>
              </sp:X509Token>
            </wsp:Policy>
          </sp:RecipientToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic256/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict/>
            </wsp:Policy>
          </sp:Layout>
          <sp:SignBeforeEncrypting />
          <sp:EncryptSignature />
        </wsp:Policy>
      </sp:AsymmetricBinding>
      <sp:Wss10 >
        <wsp:Policy>
          <sp:MustSupportRefKeyIdentifier/>
          <sp:MustSupportRefIssuerSerial/>
        </wsp:Policy>
      </sp:Wss10>
      <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
      <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
      <sp:SignedSupportingTokens>
        <wsp:Policy>
          <sp:WssX509V3Token10/>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

References

[1] K. Lawrence and C. Kaler. (2006, November).Web Service Security specification. [Online]. Available: http://www.oasis-open.org/committees/download.php/21256/wss-v1.1-spec-errata-os-SOAPMessageSecurity.htm

[2] N. Mendelsohn, D. Box, D. Ehnebuske,G. Kakivaya, A. Layman, H. F. Nielsen, S. Thatte, and D. Winer.(2007, April). SOAP Version 1.2 Messaging Framework (2nd ed.)[Online]. Available: http://www.w3.org/TR/soap12-part1/

[3] D. Box, F. Curbera, M. Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottingham, C. Riegen, and J. Shewchuk. Web services policy framework (WS-Policy), May 2003.

[4] K. Lawrence and C. Kaler. (2007, July). WS-SecurityPolicy 1.2. [Online]. Available: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html

[5] S. Ekanayake, S. Jayasoma, K. Ruwanpathirana, and I. Suriarachchi, Rampart2 presented at 9th International Information technology Conference, Colombo, Oct. 28-29, 2008.

[6] N. Gruschka, R. Herkenhöner, and N. Luttenberger, "WS-SecurityPolicy Decision and Enforcement for Web Service Firewalls". [online]. Available: http://www.diadem-firewall.org/workshop06/papers/monam06-paper-11.pdf/

[7] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In Proceedings of the 14th IEEE Computer Security Foundations Workshop, pages 82–96. IEEE Computer Society Press, 2001.

[8] K. Bhargavan, C. Fournet and A. D. Gordon, "Verifying Policy-Based Security for Web Services". Tech. Rep. MSR--TR--2004--84, Microsoft Research.

[9] A. Perrig, D. Song, and D. Phan. AGVI – automatic generation, verification, and implementation of security protocols. In 13th Conference on Computer Aided Verification (CAV),LNCS, pages 241–245. Springer, 2001.