ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

A Security Gateway for Message exchange in Services by Streaming and Validation

M.Priyadharshini[1], I.Suganya[2], N.Saravanan[3]

Research Scholar, Dept of CSE, Anna University, Chennai, India[1]

Student, Dept of CSE, Thiruvalluvar College of Engineering and Technology, Chennai, India[2]

Asst. Prof, Dept of CSE, Thiruvalluvar College of Engineering and Technology, Chennai, India[3]

Abstract: Cloud Computing is found to be today's most commonly used Service Oriented Architecture (SOA) implementation. Cloud services are exposed as Web Services which follow the industry standards such as WSDL for service description, SOAP for enabling request and response and so on. Hence Web services security is of particular importance for the security assessment of cloud systems. Securing SOAP message exchange using WS-Security standards introduces new surface for attacks. The system proposed uses a gateway at client side as well as server side which could counter the attacks introduced due to WS-Security standards as well as other attacks introduced. The handlers at the gateway uses hardened SOAP schema as the source for validating the restricted form of request according to the requestor and provider and ensures security. The SOAP Schemata which introduces loopholes in terms of security needs to be hardened, as a result of which high resource consumption leading to DOS attack is introduced. This type of attack is countered by streaming the request before the service execution at a trusted gateway and enabling earlier detection of security violations.

Keywords: Cloud Computing, Web Service Security, Streaming, Schema Validation, SOAP message, Service Oriented Architecture

I. INTRODUCTION

Web service is a functionality provided as a response to request which provides an interoperable architecture over a network. It has an interface described in a form of XML (WSDL). SOAP protocol governs the format of the web service request and response, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. WS-Security is the standard that specifies SOAP security extensions that provide confidentiality using XML Encryption standard and integrity using XML Signature standard.

SOAP messages passed as web service request could be altered in the transit by wrapping XML signature elements. Wrapping attacks is aimed at injecting a faked element into the message structure so that a valid signature covers the un-modified element while the faked one is processed by the application logic. As a result, an attacker can simulate an arbitrary Web Service request authenticating as if he is the legitimate user.

Validating incoming messages by schema validation is found to be a promising approach to counter Signature Wrapping attack. But due to flexible and permissive nature of the schemata security issues arise which are not evident at first sight. If the XML Schema used for validation is too lax, this could lead an attacker to move the signed XML contents and include content leading to un-intended effects. Hence, it becomes necessary to develop a definite, strict, loophole-free XML Schema of common Web Service messages which is called as Schema Hardening.

Schema Hardening and Validation enables signature wrapping attack to be defended whereas introduces a new type of attack Denial of Service (DOS) attack. This attack is due to the reason that the XML processing parsers consume more memory as well as processing time [5]. This could be reduced by earlier detection of security violations and reduces the time in case of intruders. The proposed system uses event based parsing (SAX) and hence streaming is done to enable this event handling in parsers [12][13].

Section II describes about the attacks and countermeasures on the XML Signature. In Section III the architecture to bring in reality a secure web service invocation by the process of validation and streaming is presented. Section IV provides implementation model that could be used for fending signature wrapping attacks and evaluation parameters that could ensure the effectiveness of the proposed work. In Section V few related works are discussed and finally in Section VI conclusion in terms of future research directions is provided.

II. ATTACKS AND COUNTERMEASURES

There are quite lot of attacks based on XML messages that could affect the security factors such as confidentiality, integrity and availability of Web Services. OASIS introduced WS-Security standard which incorporates XML Encryption and XML Signature standards for defending web services against the attacks. This measure introduced signature wrapping attacks and hence more attention is needed for avoiding un-intended effects during service invocations.

A. Signature wrapping attack

XML Encryption and XML Signature standards are applied in the SOAP message to provide confidentiality and integrity of the SOAP messages. <SignedInfo> and <SignatureValue> are two mandatory elements in XML Signature. The <SignedInfo> element includes an element Id-reference pointing to the SOAP body and a digest value computed over the referenced element. <SignedInfo> element is secured by attaching <SignatureValue> element which is the computed signature value of the <SignedInfo>element. This is typically done by a public-key algorithm such as RSA or DSA.

The SOAP message is processed in two steps: first step is to search for element specified in ID-referencing of <SignedInfo> element. The digest value of element found and compared with value in <DigestValue> element. Then in second step <Signaturevalue> value is verified against <Signed Info>.Once these processing is over function defined in the SOAP body is executed.

The attacker who eaves drops the SOAP message moves original SOAP body to the SOAP header and attaches a new SOAP body and enforces a new service and not the original one. As ID of original SOAP body remains same and passes the digest value and signature value verification. But the function execution is done based on the new SOAP body attached by the attacker, as specified in the Listing 1 where deleteUser service is replaced by upgradeRights.

```
<soap:Envelope>
<soap:Header>
    <wsse:Security>
        <ds:Signature>
            <ds:SignedInfo>
                <ds:Reference URI="#body">
                    <ds:DigestValue>....
                    </ds:DigestValue>
                </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>.....</ds:SignatureValue>
        </ds:Signature>
    </wsse:Security>
    <soap:Body wsu:Id="body">
        <deleteUser>
            <usr>John</usr>
        </deleteUser>
    </soap:Body>
</soap:Header>
<soap:Body wsu:Id="attack">
    <setAdminRights>
        <usr>John</usr>
    </setAdminRights>
</soap:Body>
<soap:Envelope>
```

Listing 1. Signature wrapping attack simulated SOAP message

To counter the above signature wrapping attack [10] [11], the schema definition can be hardened as in the Listing 2 to restrict the number of occurrences of Header and Body elements.

The hardened schema then is used for validating the message to ensure the SOAP message structure and the contents.

```
<xs:complexType name="Envelope">
<xs:sequence>
    <xs:element ref="tns:Header" minOccurs="0" maxOccurs="1"/>
<xs:element ref="tns:Body" maxOccurs="1"/>
</xs:sequence>
<xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Listing 2. Countering Schema definition for Signature wrapping attack

B. Attack Obfuscation

Attack obfuscation as such cannot be termed as an attack, where as it is a process which gives way for other attacks to be introduced [8]. Confidentiality of the attachment is ensured by encryption. The original message content is replaced with the resultant cipher text. The encrypted content may contain attacks like oversized payload, coercive parsing or XML injection. Obfuscated SOAP message will be of the form as given in the Listing 3 below

```
<soap:Envelope xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
                xmlns:ns="http://example.org/imgtrasfer">
        <soap:Body>
                <EncryptedContent>
                        AXS34mRsEXXM7ABhvXBNUINd
                        e56J6lfTPa0oUDYxPVHkm7XV8………
                        ………………………………………….
                </EncryptedContent>
        </soap:Body>
</soap:Envelope>
```

Listing 3. Attack Obfuscated SOAP message

If message validation is done after decryption it may lead system to get affected by the intended attack. If message validation is done before decryption then it may pass the message validation without identifying the attack.

Countering this attack obfuscation could be done by stepwise decryption and validation, which could lead to less memory consumption and earlier detection of intended attack.

There are few other XML attacks that could be countered by the above schema hardening and validation and stepwise handling of decryption and validation. Those attacks include coercive parsing, parameter tampering, oversized payload etc.,

III. GATEWAY FENDING SIGNATURE ATTACKS

The architecture of the system represented in Figure 1 includes the components at client side and server side. The client side components termed as client gateway does the process of signature generation, encryption and header creation. The server component at server gateway verifies the signature before which it validates the schema of request with hardened schema. Decryption process is done after the signature verification is done.

Streaming process is done on both server and client side with the help of handlers by incorporating WS-security gateway.

- The WS client creates a new SOAP message. The message is sent to the client side WS-Security Gateway.
- The client side WS-Security Gateway takes the incoming message and adds XML Signature and XML Encryption.
    - Signature and encryption is performed on the body elements or on attachments.

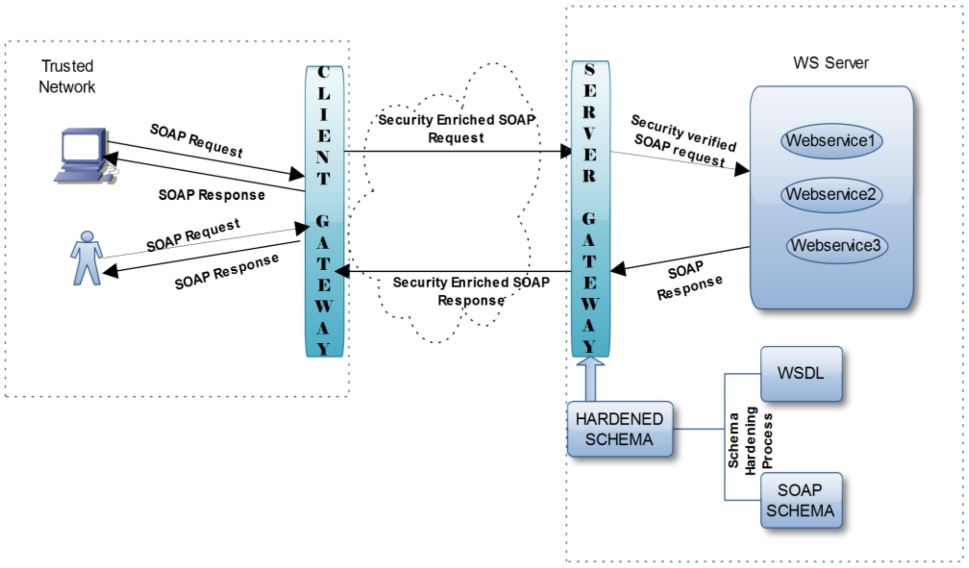


Fig 1. Architecture fending Signature wrapping attacks

- The SOAP message with newly constructed WS-Security elements is forwarded through open network to the WS server.
- The server side WS-Security Gateway receives the sent SOAP message and verifies it against Hardened SOAP Schemata. If it is valid, it is forwarded to the WS server. Otherwise, Fault is raised.
- On validation of schema further Signature Verification and Decryption of message is performed, failing the signature verification Fault is raised.
- The WS server gets the SOAP message without any WS-Security elements and hence it can concentrate on the SOAP body processing.
- After processing the SOAP message, the response is sent the same way as the request, to the WS client.

Client Side event handlers include Encryption and Signature Handler for body elements and for attachments in a sequence and Header Creation Handler, for which the input is SOAP request and output is Security enriched SOAP request as in Fig 2.

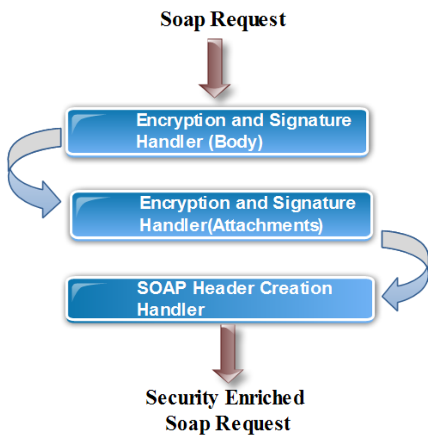

Soap Request
Encryption and Signature Handler (Body)
Encryption and Signature Handler(Attachments)
SOAP Header Creation Handler
Security Enriched Soap Request

Fig 2. Client Side Handler Sequence

The Event handlers at the server side are of prime focus in the proposed work where series of handlers are formulated and executed in sequence as listed in Fig. 3

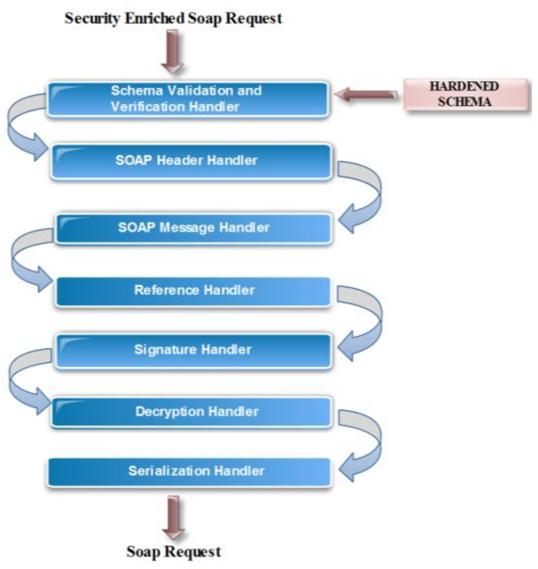


Fig 3. Server Side Handler Sequence

A. Schema validation and verification handler

The handler does validation of incoming SOAP message for correctness of schema structure and contents as per hardened schema.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

B. SOAP Header Handler

Process the added header elements and creates necessary data structure to store information needed by the other handlers. Handles XPath transformations and identifies the referenced elements for which digest values need to be calculated. For earlier detection of elements with reference id the handler maintains elements as might match and true match elements.

Might Match (possible head of result): it can be the result element, but it will be known after processing of next document elements.

True Match (head of result): the definite result of XPath expression. During the XML Signature processing, one of the assumed Might Matches always becomes the True Match

C. SOAP Message Handler

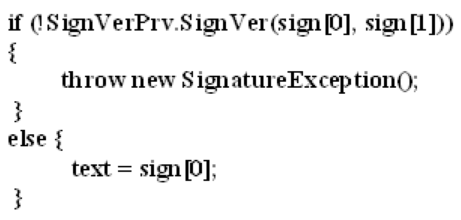
Ensures the correct ness of SOAP input parameters, as required by the provider specified in the web service description.

D. Reference Handler

Handles elements referenced with Id stored in the data structure by Header processing handler and computes their digest values.

E. Signature Handler

This Handler does verification of the XML Signature as given in the Listing 4 below



```
if (!SignVerPrv.SignVer(sign[0], sign[1]))
{
        throw new SignatureException();
 }
else {
        text = sign[0];
 }
```

Listing 4. Signature Handler Snippet

F. Decryption Handler

Decrypts the SOAP parameters and passes on to serialization Handler.

G. Serialization Handler

This handler is responsible for serialization of incoming events, reconstruction, and buffering of SOAP messages so that they can be forwarded to one of the Web Services peers.

IV. IMPLEMENTATION MODEL

A. Implementation Model

The architecture discussed in the previous section is realized for a Health Care system with a set of services from the Hospital and a set of services from insurance provider. Parsers for Schema hardening and validation are implemented using Event Handlers is done at both server and client side gateway with SAX parser enabling streaming.

Member Validation, Claim Submission, Status Checking services are created and deployed at the Insurance server and accessed by Hospital Information system. Diagnosis Details and Detailed Bill report are created and deployed at the Hospital Server and used by the Insurance Provider end as shown in Fig 4.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2013

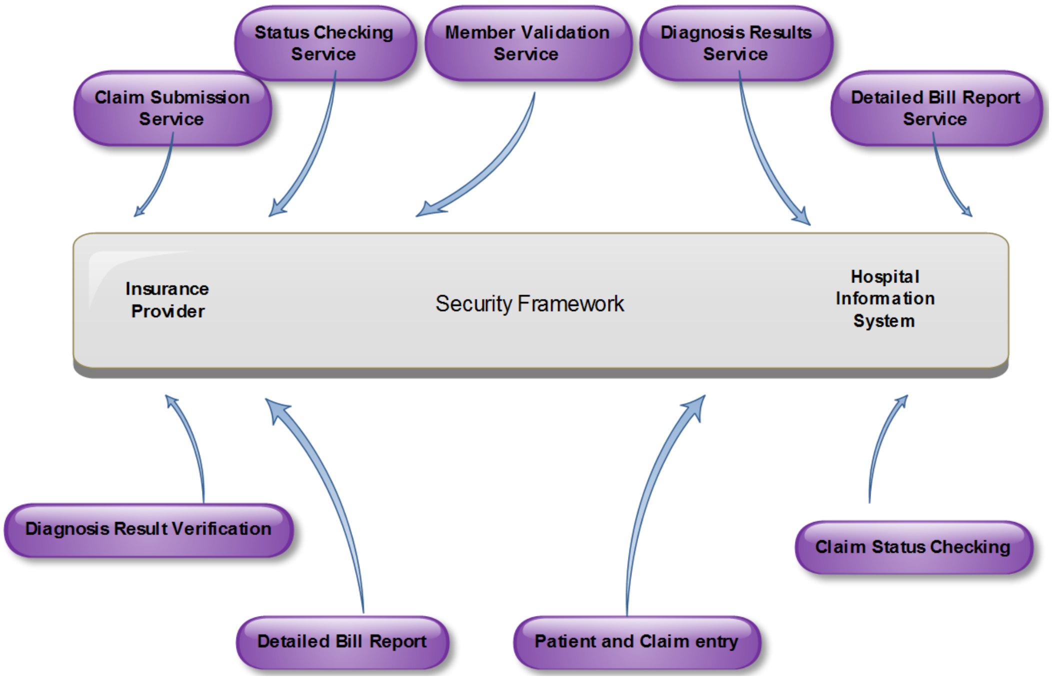
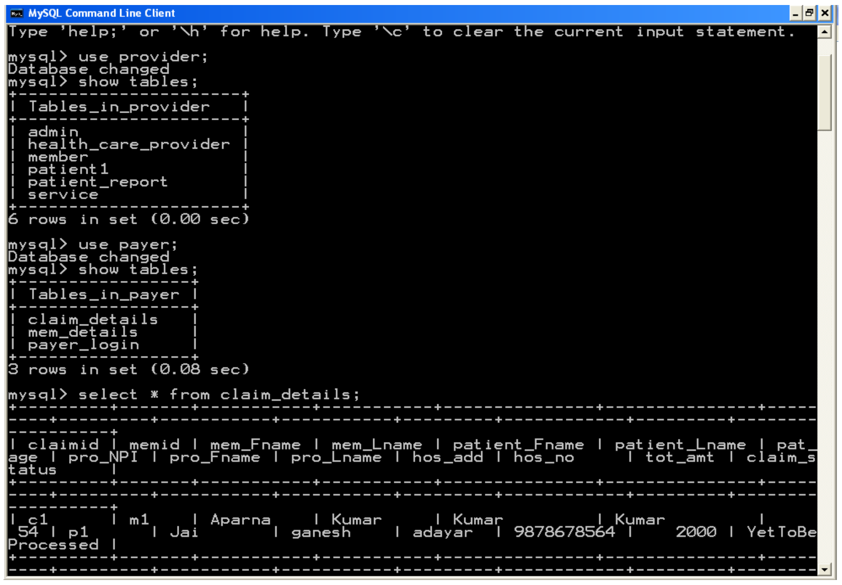


Fig 4. Implementation Model

The System is developed using JAX-WS and web applications using J2EE .

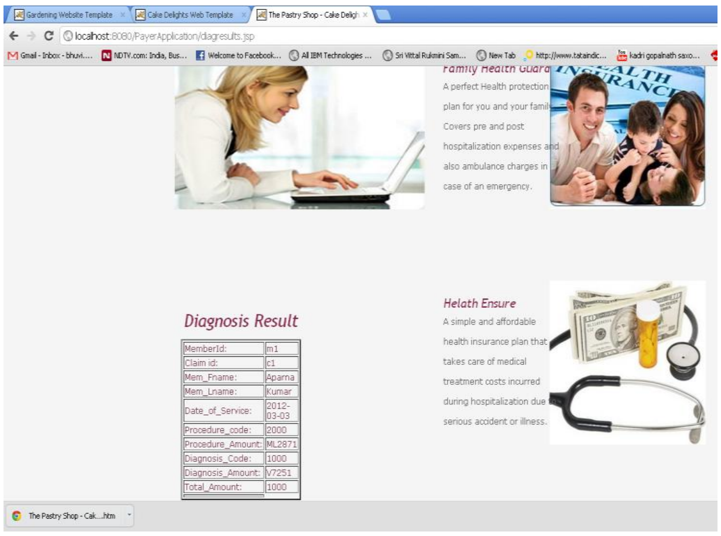
The Databases created and realised using mysql as given in Fig 5, with corresponding Hospital Server side and Insurance provider side.



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use provider;
Database changed
mysql> show tables;
+-----------------------+
| Tables_in_provider    |
+-----------------------+
| admin                 |
| health_care_provider  |
| member                |
| patient1              |
| patient_report        |
| service               |
+-----------------------+
6 rows in set (0.00 sec)

mysql> use payer;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_payer |
+-----------------+
| claim_details   |
| mem_details     |
| payer_login     |
+-----------------+
3 rows in set (0.08 sec)

mysql> select * from claim_details;
+---------+--------+-----------+-----------+--------------+------------------+-------
----+--------+-----------+-----------+---------+------------+--------+---------
---------+
| claimid | memid  | mem_Fname | mem_Lname | patient_Fname | patient_Lname | pat_
age | pro_NPI | pro_Fname | pro_Lname | hos_add | hos_no     | tot_amt | claim_s
tatus    |
+---------+--------+-----------+-----------+--------------+------------------+-------
----+--------+-----------+-----------+---------+------------+--------+---------
---------+
| c1      | m1     | Aparna    | Kumar     | Kumar         | Kumar            |
 54 | p1     | Jai       | ganesh    | adayar  | 9878678564 |   2000 | YetToBe
Processed |
+---------+--------+-----------+-----------+--------------+------------------+-------
----+--------+-----------+-----------+---------+------------+--------+---------
```

Fig 5. Database Model

The diagnosis details generated is shown as in the Fig 6 which is passed on as pdf as detailed bill report service.


Gardening Website Template
Cake Delights Web Template
The Pastry Shop - Cake Delight
localhost:8080/PayerApplication/diagresults.jsp
Gmail - Inbox - bhuvi...
NDTV.com: India, Bus...
Welcome to Facebook...
All IBM Technologies ...
Sri Vittal Rukmini Sam...
New Tab
http://www.tataindic...
kadri gopalnath saxo...
The Pastry Shop - Cak....htm

Family Health Guard
A perfect Health protection plan for you and your family
Covers pre and post hospitalization expenses and also ambulance charges in case of an emergency.

Diagnosis Result

MemberId:	m1
Claim id:	c1
Mem_Fname:	Aparna
Mem_Lname:	Kumar
Date_of_Service:	2012-03-03
Procedure_code:	2000
Procedure_Amount:	ML2871
Diagnosis_Code:	1000
Diagnosis_Amount:	V7251
Total_Amount:	1000

Helath Ensure
A simple and affordable health insurance plan that takes care of medical treatment costs incurred during hospitalization due to serious accident or illness.

Fig 6. Diagnosis Details for Detailed Bill Report Generation

B. Evaluation Parameters

Parameters used for evaluating the performance of proposed work comprises of:

- Run-time (T_r) in ms
- Memory Consumption (M_C) in terms of KBs

Schema validation is done in order to secure the data at message level. Validating the request against schema definition will increase the processing time. Signature is mainly used to secure the transfer of data which also degrades the performance. Since we can't prevent the attacks even by schema validation, we do the hardening of xml schema which again increases the processing time and degrades the performance level. Though schema validation and schema hardening degrades the performance, they are highly recommended in order to have a secured data. The performance degradation is handled by usage of streaming which could reduce the time involved in case of security violation and enables the usage of the memory and processing time for other services.

Since event based validation is performed in proposed work,

- M_C !α N_e
  where N_e is number of elements in SOAP
  Request
- $T_r \ \alpha \ (N_{ee}(T_{de} + T_{ee}) + N_{se}(T_{sv} + T_{sg}))$
  where N_{ee}, N_{se} is number of encrypted and signed elements in SOAP Request respectively,
  where T_{de}, T_{ee} is Time taken to process decryption and encryption of elements in SOAP Request
  where T_{sv}, T_{sg} is Time taken to process Sign verification and generation of elements in SOAP Request

The graph in Fig 7 shows that the memory consumption is not related to the number of elements since event based validation is performed. With respect to the run time when numbers of elements are increased the time for processing the encryption and signature gradually increases.

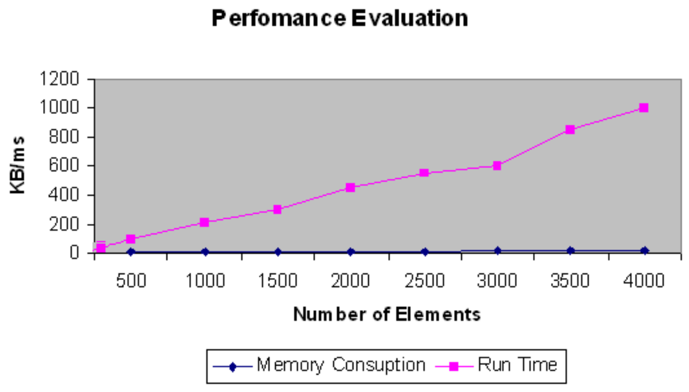


Fig 7. Performance Evaluation Graph

When compared to traditional Java XML Digital Signature API the run time and memory consumption could be considerably reduced.

V. RELATED WORKS

XPath Filtering as the signature transform XPath Filtering is developed to select complex node sets based on the XPath specification.

XPath Filter2 is designed to specify a subset of a given XML document as the information content to be signed. Since XPath Filter2 allows arbitrary XPath expressions and three filter types, it may become hard to define a correct XPath Filter expression.

Inline approach [2] is used for early detection of wrapping attacks. The weakness of the inline approach is that the SOAP Account only preserves the relationship to its parent and sibling elements.

Fast XPath [1] simpl ify and secure the use of XPath in XML digital signature, there is an even simpler subset of XPath, called FastXPath. FastXPath contains only forward directions. FastXPath allows for fast single-pass SAX style selection of the elements to process. On the other hand, FastXPath turns out to be best possibly resistant to signature wrapping attacks. As it is required to explicitly name every single element on the path from document root to the signed subtree, there is no flexibility here to move any signed contents to another location within the document.

However, even with FastXPath the threat of signature wrapping attacks cannot be averted completely. Though FastXPath poses severe restrictions to the abilities of defining a signature reference, this turns out to be a trade-off between security and flexibility, as every flexibility within the reference can potentially be exploited for a wrapping attack.

Hence FastXPath Referencing becomes a optimal solution. But the main drawback is that the FastXpath filtering also cannot prevent this xml signature wrapping attack completely and hence other techniques like schema validation [4] and schema hardening need to be used. Also streaming enhances the performance of the security restriction validations [3] [9].

Few Frameworks were also devised concentrating on the security aspects discussed. Nils Gruschka, et.al., discussed a suitable means to prevent web services attacks is the full grammatical validation of messages by an application level gateway before forwarding them to the server[12]named as “Check way”. The Check Way WSDL Compiler gets the Web Service server’s Web Service description, generates the corresponding XML message Schema, ”hardens” the description, and advertises the modified description to a Web Service client. The Check Way Gateway validates all SOAP messages against the Schema, forwards the message if it is valid, and rejects the message if it is not valid. Parsing and Validating of XML documents works entirely based on SAX interface. Hence, Checkway can process very large documents easily. Inspite of this, validation of SOAP message by application level gateway could not resist web services attack totally and also attachments are not handled and works only for atomic services.

Meiko Jensen, et al., analyzes the effectiveness of the specific countermeasure of XML Schema validation in terms of fending Signature Wrapping attacks and investigates the problems of XML Schema validation for Web Services messages, and discusses the approach of Schema Hardening, a technique for strengthening XML Schema declarations [11]. Validating incoming messages by schema validation may seem to be a promising approach to fight Signature Wrapping but brings huge performance lag.

In order to process both XML encryption and XML signature simultaneously, Nils Gruschka, et al., introduces Stream-based WS-Security processing system which enables a more efficient processing in service computing and increases the robustness against different types of Denial-of-Service (DoS) attacks[13]. The introduced engine is capable of processing all standard-conforming applications of WS-Security in a streaming manner, but handling of attachments and composition is not performed. Handling SOAP attachments is addressed in the proposed system leaving composition part un implemented.

VI. CONCLUSION AND FUTURE DIRECTIONS

The conclusion that can be drawn from this work is that the XML Schema hardening can only be one brick towards Signature Wrapping resistant documents. With schema validation alone it cannot undoubtedly be ensured that Signature Wrapping is completely avoided, but it raises the bar for potential attackers and reduces the attack surface demonstrably. Hence the concept of schema hardening has to be applied to all included namespaces.The streaming process done here could be used for fending attack obfuscation in case of atomic services and services with attachments, future work can focus on the composite services.

REFERENCES

1. Gajek .S, Jensen .M, Liao .M, and Schwenk .M, “Analysis of signature wrapping attacks and countermeasures,” in the proceedings of International Conference on Web Services, pp. 575–582, 2009.
2. Gajek .S, Liao .L, and Schwenk .L, “Breaking and fixing the inline approach,” in the Proceedings of the 2007 ACM workshop on Secure web services,. ACM Press, pp. 37–43, 2007.
3. Gruschka .N, Jensen .M, and Iacono .L, “A Design Pattern for Event-Based Processing of Security-Enriched SOAP Messages”, in the Proceedings of Second International Workshop on Security Aspects in Grid and Cloud Computing, pp. 410- 415, 2010.
4. Gruschka .N and Lo Iacono .L, “Vulnerable Cloud: SOAP Message Security Validation Revisited,” in ICWS ’09: Proceedings of the IEEE International Conference on Web Services, pp. 625 – 631, 2009
5. Gruschka .N and Luttenberger .N, “Protecting Web Services from DoS Attacks by SOAP Message Validation”, in Proceedings of the IFIP International Federation of Information Processing, Vol.201, pp.171 – 182, 2006.
6. Gruschka .N, Luttenberger .N, and Herkenhoner .R, “Event-based SOAP Message Validation for WS-SecurityPolicy-enriched Web Services,” in Proceedings of the 2006 International Conference on Semantic Web & Web Services, 2006.
7. Imamura .T, Clark .A, and Maruyama .H, “A Stream-Based Implementation of XML Encryption,” in the Proceedings of ACM Workshop XML Security (XMLSEC ’02), pp. 11-17, 2002.
8. Jensen .M, Gruschka .N, and Luttenberger .N, “The Impact of Flooding Attacks on Network-Based Services,” in the Proceedings of the Third International Conference on Availability, Reliability and Security (ARES ’08), pp. 509-513, 2008.
9. Lu .W, Chiu .K, Slominski .A, and Gannon .D, “A Streaming Validation Model for SOAP Digital Signature”, in the Proceedings of the 14th IEEE International Symposium High Performance Distributed Computing (HPDC ’05), pp.243 – 252, 2005.
10. McIntosh. M and Austel .P, “XML signature element wrapping attacks and countermeasures,” in the Proceedings of the workshop on Secure web services, ACM Press, 2005, pp.20–27, 2005.
11. Meiko Jensen, Christopher Meyer, Juraj Somorovsky, and J¨org Schwenk Chair for Network and Data Security “On the Effectiveness of XML Schema Validation for Countering XML Signature Wrapping Attacks” in the International Workshop on Secured Services in the Cloud, IWSSC, pp. 7 -13, 2011.
12. Nils Gruschka, Meiko Jensen, Luigi Lo Iacono, and Norbert Luttenberger, “Server-Side Streaming Processing of WS-Security”, in the IEEE Transactions On Services Computing, Vol. 4, No. 4, October-December 2011.
13. Somorovsky .J, Jensen .M, and Schwenk .J, “Streaming-Based Verification of XML Signatures in SOAP Messages”, in the Proceedings of the 2010 6th World Congress on Services (SERVICES ’10), pp.637 – 644, 2010.