IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

A New Partitioning Algorithm to Enhance Performance of SOAP Message Security Processing

Nidhi Arora[1], Savita Kolhe[2], Sanjay Tanwani[3]

Assistant Professor, Department of Computer Science, M.B. Khalsa College, Indore, India[1]

Senior Scientist (Computer Applications), ICAR-Directorate of Soybean Research, Indore, India[2]

Professor and Head, School of Computer Science and IT, Devi Ahilya University, Indore, India[3]

Abstract: Web Service Security (WS-Security) processing suffers from performance bottleneck. With large data centric web services and hardened schema it becomes even worst. A new partitioning algorithm with parallel stream based security processing model is developed for large Simple Object Access Protocol (SOAP) messages. The aim is to partition SOAP message and to distribute the processing load on multiple cores in order to enhance the performance of SOAP message security processing. There are several steps involved in this approach but SOAP message partitioning is mainly focused in this paper. Since the efficiency of SOAP security processing depends on both Extensible Markup Language (XML) data size and its structure, these factors are considered while developing partitioning algorithm. Each partitioned SOAP message data is uniformly distributed to parallel running instances of stream based security processor so that the load among cores can be balanced. It is observed from the results that the implementation of new partitioning algorithm with parallel stream based WS-Security processing model has significantly improved the performance of SOAP message security processing. The fundamental approach presented in this paper is useful for efficient security processing of large SOAP messages.

Keywords: Web Services, WS-Security, SOAP message processing, XML security, XML data partitioning.

I. INTRODUCTION

Web services use Simple Object Access Protocol (SOAP) to exchange messages between two endpoints over Hyper Text Markup Language (HTTP). It is vital to secure these SOAP messages when services are sharing critical data like money or company's confidential data. Web Services Security (WS-Security) with its underlying standards XML- Encryption and XML- Signature is widely adopted to provide security of SOAP based web services [1].

Size and complexity of the SOAP message affects the performance of WS-Security [2], [3]. If the web service itself is sharing large amount of data, the memory and CPU requirements become high. WS-Security processing increases the size of message after processing [3], [4], as it adds several tags like <xenc:EncryptedData>, <ds:SignatureValue> and other supporting tags in the original SOAP message after security processing. Also, it involves complex processing like cipher calculations.

Therefore it requires more CPUs cycles, memory and bandwidth as compared to non-secured SOAP message processing [2-4]. This makes the web service slow, unresponsive and even vulnerable to several attacks like DoS, DDoS [5], [6] etc. Security attacks eventually fill the server's buffer space by continuously sending multiple requests or extremely long messages to it. Once the buffer memory is full, no further connections can be made to the server. Also, it can crash the targeted server making the service unavailable [5-7]. Security techniques like schema validation and use of hardened schema are sufficient counter measures [6-11] to prevent web services from several attacks specially XML signature wrapping attack. SOAP messages are validated against strict schema definition and strictly prohibit any other content that is not contained in the hardened XML Schema. But these techniques suffer from performance bottleneck [10], [11]. This paper presents a new SOAP message partitioning algorithm with schema analyzer algorithm in order to distribute processing load of WS-Security processor to parallel instances running on multiple cores. Data parallel model for stream based processing of secured web services is also developed [12].

The model has various processing components like Parser, Schema Validator, Encryption/Decryption and Signature/Verification component. Schema validation and Hardened schema is used in order to prevent web services from aforesaid attacks. Before security processing, SOAP message is divided into parts using the new partition algorithm with an aim to enhance the performance of WS-Security processing and schema validation with hardened schema. Since the efficiency of SOAP message containing XML data depends on both XML data size and its structure [13], [14], these factors have been considered when XML data is partitioned. The development of algorithms is discussed in detail in the paper. Experiments are conducted on different sizes of SOAP messages to evaluate the performance of parallel stream based model on various hardware platforms using the new partitioning algorithm.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

II. METHODOLOGY

Data centric SOAP messages generally have repetitive structure of elements. Example of SOAP message containing repetitive <dept> elements containing official information of different departments is shown in Fig. 1.

```
<?xml version=" 1.0" encoding= "UTF-8" ?>
<SOAP:Envelope xmlns:SOAP=" http://www.w3.org/2003/05/soap-envelope" >
<SOAP:Header></SOAP:Header>
<SOAP:Body>
<totals xmlns=xmlns=" http://www.wiley.com/SOAP/accounting">
<dept id=" 1" >
<gross>50000</gross>
<net>20000</net>
</dept>
<dept id=" 2" >
<gross>90000</gross>
<net>40000</net>
</dept>
</totals>
</SOAP:Body>
</SOAP:Envelope>
```

Fig.1. Sample SOAP message

These repetitive elements are distributed in different partitions of SOAP message. SOAP message partitions are constructed at client side according to the schema definition of SOAP message. Schema information i.e. structure of SOAP message is usually available with SOAP message; otherwise it can be derived from WSDL [15]. Partitioning divides the large SOAP message into nearly equal size small partitions. Size of SOAP message is obtained from header information [16]. This information is used to equally distribute workload among different cores using new partitioning algorithm.

The main objective of this algorithm is to optimize partition size which balances the load of parallel parser instances, running on different cores. This is done with few heuristic rules derived from schema definition. Another objective is to ensure zero or limited communication between partitions. For this load balancing is done statically i.e. partitions are made before the parsing process begins.

III. PARTITIONING ALGORITHM

Partitioning algorithm consist of several modules as shown in Fig.2. These are

- SOAP message Pre-Partitioning
- Schema Analyzer
- XPath Resolver
- Partition wise XPath Creator
- SOAP message Preprocessing
- Dummy Creator

A. SOAP Message Pre Partitioning

SOAP message contains several tags starting with "<" and ending with ">" and having elements according to the schema definition. First SOAP message is partitioned randomly into equal size parts.

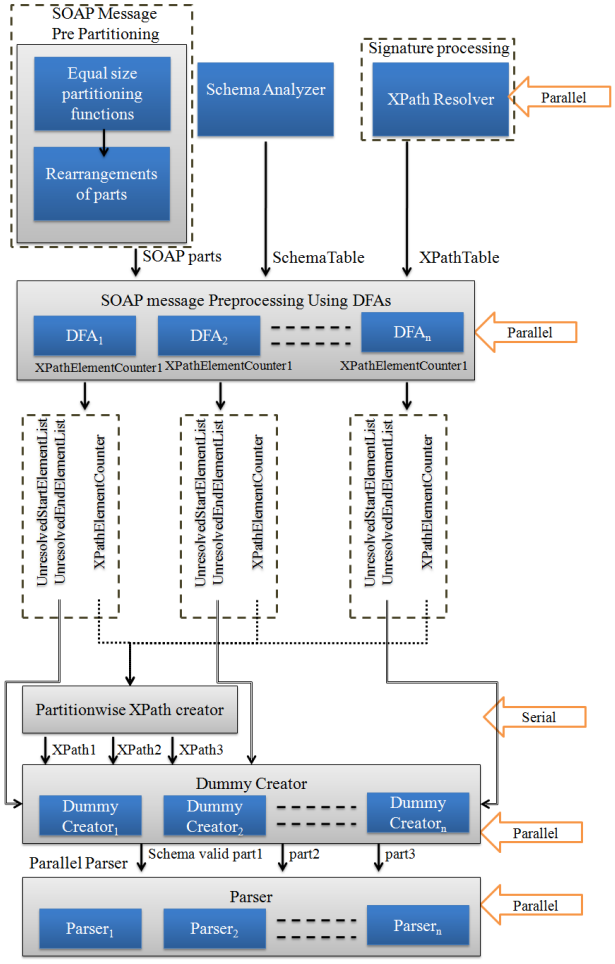


Fig.2. Different modules used in partitioning algorithm

Size of partition is calculated using the following formula:
Size of partition = Size of SOAP message

Number of partitions

Parts are then rearranged taking care that every part start with "<" or "</xxx>".

B. Schema Analyzer

The output of partitioning algorithm should be well-structured schema valid partitions; otherwise parser will reject those partitions. Therefore, information derived from schema analyzer is used in our partitioning algorithm to divide SOAP message into well-structured and approximately equal parts.

The Schema Analyzer Algorithm (Fig. 3) is developed to analyze basic structure and list of elements in SOAP message from schema definition. Hardened schema definition [6][10] is used which is devoid of unbounded and xs:any elements. It eliminates all extension points and weak definitions in the SOAP Schema [6] in order to protect web services from XML signature wrapping attacks.

Information like hierarchy of elements, optional and must elements etc. are obtained from schema analyzer algorithm and schema table is generated as output. An example of SOAP message schema definition is described in Fig. 4.

DOI 10.17148/IJARCCE.2016.5134

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

Algorithm 1 (XML Schema Analyzer)
Input: XML Schema of SOAP message/document.
XML Schema file. //Schema file derived from WSDL for SOAP can be set as default.
Output: List SchemaTable containing nodes entries about attributes of each element i) ElementName ii) ParentID iv) DefaultValue iii) Size iv) MaxOccurs v) MinOccurs
Temporary: CurrentParentID= -1 //CurrentParentID variable contains the information about CurrentParent
CSize=0 //Size info of current processed element
I. Initiate the list SchemaTable with an empty list.
II. Read schema definition one node / element at a time. //This can be using DOM or SAX parser.
III. For each element in schema definition
a. If start element tag
i. If start of complex element tag, Set
- SchemaTable→ElementName= value of attribute name specified in tag
- If CurrentParentID==-1
- SchemaTable→MinOccurs=1 else
- SchemaTable→ MinOccurs = value of attribute minOccurs specified in tag
- CurrentParentId= IndexofElement
- SchemaTable→ParentID=-1
- SchemaTable→Size=0

ii. If start of simple element tag, Set
- SchemaTable→ElementName= value of attribute name specified in tag
- SchemaTable→maxOccurs = value of attribute maxOccurs specified in tag
- SchemaTable→DefaultValue= value as per attribute type specified in tag*
- SchemaTable→Size=value of attribute type specified in tag
- Update size information of all its roots by adding size of current with size information of roots recursively.

b. If end of element
i. If end of complex tag
ii. Set CurrentParentID=SchemaTable→ParentID at location CurrentParentID

IV. Repeat step II to III for every element till end of schema file

*// information according to attributes values specified in element tag.
//Default Values are created as per type and constraints on it.
If type=="string", set DefaultValue="dummy"
If type=="decimal", set DefaultValue=0
If type="Positive Integer", set DefaultValue=1

Fig.3. Schema Analyzer for partitioning SOAP message into schema valid parts

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

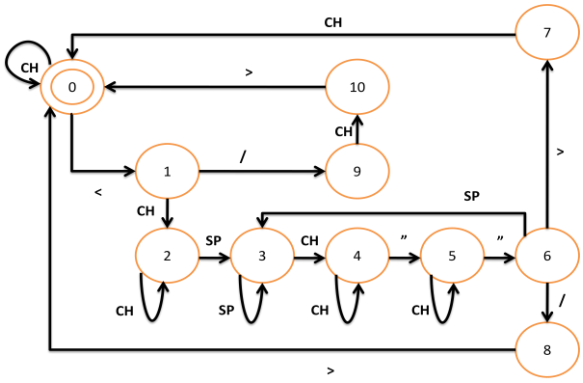
Fig.4. Part of official schema definition of SOAP message

C. XPath Resolver
XPath or ID is used commonly to refer to the elements that are to be signed or encrypt. But it makes web services vulnerable to XML Signature wrapping attack [10]. FastXPath is a faster and secure way to refer to signed elements [10], therefore it is used in current work. XpathResolver module is developed and used to resolve FastXPath expressions prior to partitioning. XPath table is created as output which is used by Partitionwise XPath Creator module.

D. SOAP Message Preprocessing
Algorithm finds start and end elements of each partition as per the Schema definition. These partitions are distributed to parallel running Deterministic Finite Automaton (DFA) for preprocessing. DFA as shown in Fig. 5 has been used for performing this task. These partitions cannot be passed directly to the parser as it is not known from which state to begin parsing each part. The DFA finds the start and end state of each part.

Results of this module are used by Dummy Creator in order to find out incomplete elements and tag pairs.

This module also creates list of namespace used in the SOAP message which is further utilized by Exclusive SAX Canonicalization [17]. Canonicalization is a process performed during signature processing by security processing module of parser.



CH= Any unicode character except <, >, /, !, ", '
SP=Blank Space

Fig.5. DFA used for partitioning SOAP message in well-formed parts

E. Partitionwise XPath Creator
New XPath expressions are created for each partition, as XML elements that are to be signed, are distributed among partitions. Their positions are changed from the position obtained from original XPath expression. Partitionwise XPath contains relative position of these elements in partitions.

F. Dummy Creator
SOAP message partitions may have incomplete elements or schema invalid parts because some of sequence elements may be distributed among parts. In order to make these parts schema valid, some dummy XML tags need to be appended at start and end of each part. Schema table obtained from Schema Analyzer Algorithm is used to create dummy tags for getting well-formed SOAP message partitions. This is done using Dummy Creator module that creates dummy tags for each part according to schema definition and appends them accordingly.
These partitions are processed further using parallel instances of parser on different cores in parallel manner. Parser has security processing as one of its component. Parallel Stream based Model for WS-Security [12] is used to process these parts and apply security to SOAP messages. After processing these partitions are merged and sent to server along with information on partition count and size of each partition.

IV. RESULTS AND DISCUSSION

The performance evaluation of partitioning algorithm with parallel stream based SOAP message security processing model is done on single core, dual core, core i5 and i7 systems. Experiments are conducted for SOAP message of varying sizes viz. 4MB, 8MB, 16MB, 32MB and 64 MB. The observation on processing time for parallel security processing is noted for each size on each platform

separately. Each test case is run 50 times to skip worst case observations, the average of all observations are taken. Results are observed to know the advantage of partitioning on SOAP message security processing.

The processing time required by secured SOAP message (64 MB) on different hardware platforms after partitioning is depicted in Fig. 6.

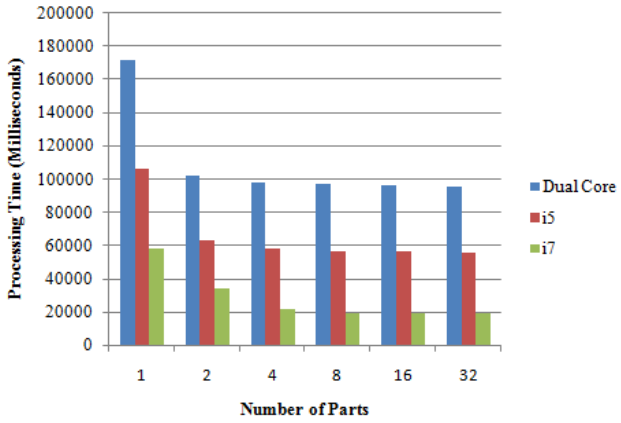


Fig.6. Processing Time for Secured SOAP Message (64 MB) after Partitioning

Running 64 MB SOAP message (without partitioning) on dual core requires 172131 ms processing time. When message is divided in 32 parts, performance is improved up to 44%. Similarly on i5, running the same SOAP message as a whole takes 106158 ms. Performance improvement is observed as 48% after dividing it into 32 parts. On i7, after dividing 64 MB SOAP message into 32 parts, performance is significantly increased up to 68%.
It is observed from the results that increasing number of parts enhances the performance but as number of parts becomes higher than number of cores available, there is negligible gain in performance and it reaches to constant value. Therefore, the performance improvement of parallel security processing depends on the number of partitions as well as the number of cores available.

The results of the approach used in the present work shows nearly linear scalability at least up to 64MB of SOAP message when the data size changes. Therefore, the fundamental approach presented in this paper can be widely accepted for efficient security processing of large SOAP messages.

V. CONCLUSION

A new partitioning algorithm is developed for partitioning large SOAP messages containing XML datasets in order to distribute the security processing load on parser among different cores. Number of partitions used in parallel processing improves the performance of security processing. Performance of WS-Security processing can be enhanced by using good partitioning techniques with a reasonable number of partitions. The partition algorithm is applied prior to parsing and security processing. Therefore it can also be used for any other parser model or configuration.

REFERENCES

[1] N. Gruschka, M. Jensen, and L. Iacono, “A Design Pattern for Event-Based Processing of Security-Enriched SOAP Messages,” In Proceedings of Second International Workshop on Security Aspects in Grid and Cloud Computing, pp. 410- 415, 2010.
[2] S. Makino, K. Tamura, T. Imamura, and Y. Nakamura. “Implementation and performance of WS-Security”, IBM Research Report, Tokyo Research Laboratory, 2007.
[3] Robert, E.van and Z. Wei, “An Overview and Evaluation of Web Services Security Performance Optimizations,” In IEEE International Conference on Web Services, 2008.
[4] T. Imamura, A. Clark, and H. Maruyama, “A stream-based implementation of XML Encryption,” In Proceedings of ACM workshop on XML security, New York, NY, USA, pages 11–17, 2002.
[5] M. Jensen, N. Gruschka, R. Herkenh¨oner and N. Luttenberger, “SOA and Web Services:New Technologies, New Standards – New Attacks,” In Proceedings of the 5th IEEE European Conference on Web Services, 2007.
[6] M. Jensen, C. Meyer, J. Somorovsky, and J¨orgSchwenk, “On the Effectiveness of XML Schema Validation for Countering XML Signature Wrapping Attacks” in the International Workshop on Secured Services in the Cloud Chair for Network and Data Security, IWSSC, pp. 7 -13, 2011.
[7] M. Ibrahim B and M. Shanavas A R, “Constructing Solutions to SOA Attacks on SOAP Web Services-A literature Review”, In International Journal of Scientific Engineering and Technology, Vol. 3 no 3, pp 564-569, 2014.
[8] Thomas, “DDoS defense system for web services in a cloud environment”, http://dx.doi.org/10.1016/j.future.2014.03.003, Future Generation Computer Systems 37–45, Elsevier, 31-39, 2014.
[9] M. Ibrahim AK, L. George, K. Govind and S. Selvakumar, “Threshold Based Kernel Level HTTP Filter (TBHF) for DDoS Mitigation, Computer Network and Information Security,” Published Online in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijcnis, 2012.
[10] C. Mainka, “Automatic Penetration Test Tool for Detection of XML Signature Wrapping Attacks in Web Services”, Ph.D. dissertation, Ruhr-Universität Bochum, 2012.
[11] C. Mainka, M. Jensen, L. Lo Iacono, and J. Schwenk, “XSpRES: Robust and Effective XML Signatures for Web Services,” In 2nd International Conference on Cloud Computing and Services Science, 2012.
[12] N. Arora, S.Kolhe, S. Tanwani, “Parallel Stream Based Processing Model for WS-Security”(Communicated)
[13] H. Kurita, K. Hatano, J. Miyazaki, and S. Uemura, “Efficient Query Processing for Large XML Data in Distributed Environments,” In Proceeding of 21st International Conference on Advanced Networking and Applications (AINA), 2007.
[14] R. Bordawekar, L. Lim, and O. Shmueli, “Parallelization of xpath queries using multicore processors: challenges and experiences,” In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pages 180–191, 2009.
[15] N. Gruschka, M. Jensen, T. Dziuk, “Event-based Application of WS-SecurityPolicy on SOAP Messages”, ACM, USA, 2007.
[16] A. Nasridinova , “A Study on Detection Techniques of XML Rewriting Attacks in Web Services,” In International Journal of Control and Automation, Vol.7, No.1, pp.391-400, 2014.
[17] J. Somorovský, “Streaming-based Processing of Secured XML Documents,” Ph.D. dissertation, University Bochum, 2009.