IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 1, January 2016

Parallel Stream Based Processing Model for WS-Security

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 Services are widely adapted for integrating heterogeneous information systems in a cost-efficient way. Simple Object Access Protocol (SOAP) messages are standard way to exchange the information between web services. Web Services Security (WS-Security) specification is used to secure SOAP messages but adds significant overhead on SOAP message processing. It increases the size of the message on the wire. Also web services are vulnerable to several attacks. Processing efficiency and robustness against certain attacks are important issues of web services security. Schema validation and Hardening are the promising methods to prevent web services from such attacks but have performance bottleneck. A new Parallel Stream based Security Processing model has been developed in order to enhance the performance of WS-Security processing and to secure web services from several attacks. A new methodology is described in the paper in which large SOAP messages are partitioned into schema valid parts. Individual parts are distributed to parallel instances of security processors running on multiple cores in order to reduce the processing load. Experiments are conducted on different sizes of SOAP messages, various security patterns and respective processing time for each is analyzed. Analysis of results reveals that the new parallel stream based security processing model has shown significant improvement in performance as compared to the serial processing.

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

I. INTRODUCTION

A Web Service uses Simple Object Access Protocol (SOAP) to exchange messages between two endpoints over Hyper Text Transfer Protocol (HTTP). It is vital to secure these SOAP messages when services are sharing critical or confidential data. SOAP messages can be altered in the travel by wrapping XML signature elements. The security attacks like DoS attack, XSW and flooding attack etc. can spoil the sensitive data. It can crash the targeted server making the service unavailable [1]. The current specifications available for web services security do not consider the problem of availability [2].

WS security protocol stack is used to apply security to web services and Web Services Security (WS-Security) is an important component of it. The primary focus of WS-Security is XML Encryption and XML Signature [3]. WS-Security processing incurs lots of memory and processing time [4] [5]. It also provokes new kind of DoS attacks. Security processing with Schema Validation [5][6] and Schema Hardening [7] are found to be promising approaches to prevent DoS, XWA attacks. Parsing SOAP message with schema validation can detect any invalid or harmful messages and therefore can be rejected before giving it to application. This can then protect the system from attacks like DoS and XWS done unintentionally or maliciously, without causing error in WS systems [8]. In most of the cases, schema validation is avoided because it is a highly time consuming and resource intensive process [1] [4] [9]. Security processing itself requires complex computations. The overall processing time goes up steeply, if schema validation is also to be carried out. Also, WS-Security adds significant overhead to SOAP-processing due to the increased size of the message on the wire [10]. In an environment where users are numerous and data is huge, it becomes critical for web services to communicate with optimum performance and efficiency [11]. Existing security techniques are not able to detect such attacks effectively and efficiently.

A reliable and efficient security solution is required in order to protect web services from aforesaid attacks. To handle these issues, data parallel model for stream based encryption and decryption processing of web services is developed. Stream based processing of security [6] [9] [12] enhances the performance and early detection of attacks. A new SOAP message partitioning algorithm [14] is used in order to distribute the load among parallel running instances of WS-Security processor. Parallel processing [13] is used in order to enhance the overall performance of security processing.

The paper describes the design, working and implementation of the parallel stream based WS-Security processing model. The performance of new model is tested for on different sizes of SOAP message and various security patterns. The results of new model are compared with respect to the existing serial model. Analysis of results reveals that new model has improved performance of encryption significantly over existing serial model.

II. PARALLEL STREAM BASED WS-SECURITY PROCESSING MODEL

Parallel stream based WS-Security processing model provides a solution for the problems with evaluation of large secured XML documents, mainly SOAP messages. It

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 1, January 2016

provides a parallel streaming-based Web Services Security Gateway as shown in Fig. 1. The Client End Web Services Security Gateway is used for processing of SOAP messages containing XML encryption whereas The Server End Web Services Security Gateway is used for processing of SOAP messages containing XML decryption.

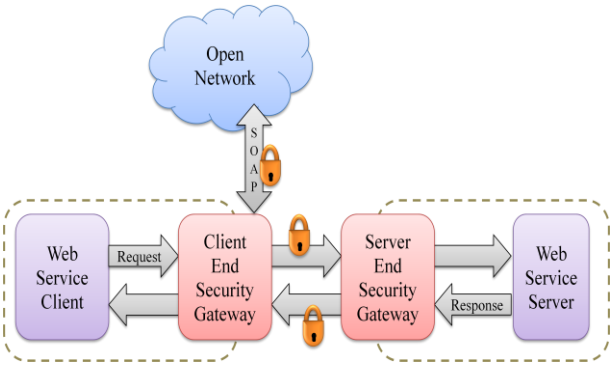


Fig1. Secured SOAP message processing using WS security gateway

The whole processing of XML encryption and decryption is done with a combination of streaming-based approach and parallel processing as shown in Fig. 2. The message is first passed to SOAP message partitioning module [14] in order to get equal size schema valid partitions. Data centric SOAP messages generally have repetitive structure of elements. These repetitive elements are distributed in different partitions of SOAP message / document. Then each part is parsed in parallel on multiple cores. Streaming processing through Simple API for XML (SAX)[15][16] is used which processes SOAP message step-by-step and allows interrupting the parsing as soon as an invalid element is encountered [5]. Validation of incoming messages by schema validation is used as it is promising approach for early detection of security violation. Hardened schema is used here in order to protect SOAP message from DoS, XWA etc. The security processor is added to the pipeline of stream based parser. It checks whether data requires security processing. If requires, then security processor performs the security related processing on that data.

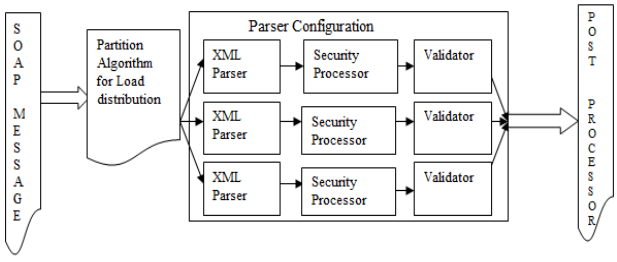


Fig.2. Parallel stream based security processing model for WS-Security

The element name that is to be encrypted is derived from the security policy before parallel parser instance creation [14]. Relative positions of elements after partitioning is also calculated and added as header element of partitions. If due to partitioning, element is divided and distributed across partitions, distributed element parts in each partition are encrypted separately. Encryption algorithm and parameters are also derived prior to parsing and encryption/decryption key is buffered.

A. Client End Security Processing

The communication is initiated by client for requesting the access to a specific WS from the server. At the time of service request, client sends SOAP request to server. The client ensures that messages are in the proper syntax and conform to all security measures required by the server.

Client prepares the SOAP message using client end web service security gateway by applying encryption on whole/part of SOAP message. This is done in our model according to the rules specified in the negotiated security policy of end points. Message has one or more encrypted parts. Secured SOAP message is then transferred to ultimate receiver travelling through none or a number of intermediaries.

1] Encryption Module:

During parsing, data is transmitted from one parser module to another parser module via XML events. Security processor is used as an encryption module at client end security gateway shown in Fig. 3.

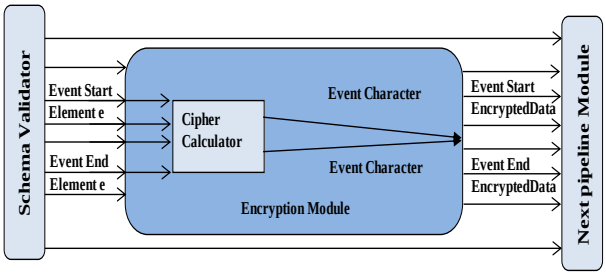


Fig.3. Security processor as encryption module

If an XML element event name is matched with element name to be encrypted, the encryption module encrypts the incoming stream. It creates a new series of XML events for <EncryptedData> element and its sub-elements. These are then pushed to the next module of the pipeline. First, all events from <EncryptedData> to <CipherValue> are sent to the pipeline. Then, incoming events are redirected to a cipher calculator to calculate the cipher of event stream until the end tag of the specified element is found. The cipher calculator then converts cipher texts to character chunks and then pushes as character events to the next module. Lastly, the </CipherValue> and </EncryptedData> are created and pushed to next module of the pipeline.

After processing of all parts, these partitions are combined in same order as were in original SOAP message. SOAP Account [17-19] is updated with two information –(i) total number of parts and (ii) size of partitions. This is done so that they can be processed according to the updated information in SOAP Account. Message is then passed to the server where server end security gateway will perform the security processing before it is being used by application.

B. Server End Security Processing

Server End Web Services Security Gateway receives SOAP message and divides message into partitions according to the information in SOAP account. Then it

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 1, January 2016

decrypts message partitions in parallel by creating instances of stream based parser and assigning them to separate cores. Decrypted message is then validated according to schema definition. After validation, message partitions are post processed to generate the original message these are then transferred to application. In general, the validation of SOAP message according to the schema definition may or may not be done, as it depends on the negotiated end point configuration. In our model validation is used as compulsory component of secured SOAP message parsing.

2] Decryption Module:
Security processor is used as a decryption module at server end security gateway as shown in Fig. 4.

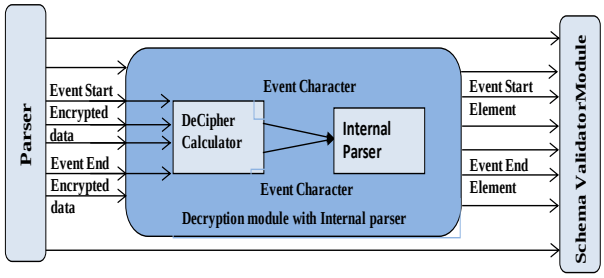


Fig.4. Security processor as decryption module

The decryption module uses a buffer that collects all incoming character chunks that occur between the start tag <CipherValue> and end tag </CipherValue>. When the decryption module detects the </CipherValue> end tag, the decryption module decrypts the buffered data. Decryption module uses a subsequent internal parser module. It resides virtually within event pipeline module. Its task is to parse the decrypted output that it retrieves from the decryption module and generates events according to decrypted data and passes them to the next module.

III. RESULTS AND DISCUSSION

Experiments are carried out to evaluate the performance of the new model and compared with other XML security models based on stream based processing. Experiments are performed on single core, dual core, core i5, and i7 systems.

XML containing personnel information as shown in Fig. 5 is used for testing the model.
Format of SOAP XML elements used are as follows:

- Element-centric XML data with large number of sibling nodes.
- Element-centric deeply nested XML elements.
- Attribute-centric XML data.

Following encryption patterns are used with test data for encryption of:

- Data enclosed in SOAP element.
- Whole SOAP element.
- Whole SOAP message/document.
- Encrypted element.

The impacts on processing speed are evaluated with experimental data on three factors:

```
<SOAP message body here><phonebook>
    <person>
        <firstname>Samta </firstname>
        <lastname>Daftari</lastname>
        <address>...</address>
        <phone>123</phone>
        <mobile>123</mobile>
    </person>
    <person>
        <firstname>Savita</firstname>
        <lastname>Kolhe</lastname>
        <address>...</address>
        <phone>345</phone>
        <mobile>345</mobile>
    </person>
    ...
    ...
</phonebook>
```

Fig.5. Part of XML data having repetitive structure

- Size of the text to secure viz. 4MB, 8MB, 16MB, 32MB and 64 MB.
- Proportion of security text to whole XML message.
- Number of partitions (2, 4, 16, 32) used to parse in parallel.

The results of parallel processing of Encryption and Decryption with validating parser on i7 are depicted in Figures 6 and 7 respectively. X-axis shows the number of threads running in parallel. Numbers of threads running are equal to number of partitions, where each partition is assigned to a different thread. The processing time (milliseconds) for parsing and encryption/decryption of different threads running in parallel are depicted on the Y axis.

It is clear from these figures that the processing time required for encrypting/decryption XML data without partitioning for all sizes of file is higher as compared to dividing XML data in number of parts (2, 4, 8, 16 and 32) and running individual parts on different cores in parallel. The encryption/decryption processing time reduces gradually when the data is partitioned in two, four and eight parts. However, there is a small change in processing time when the file is partitioned further (16 and 32 parts).

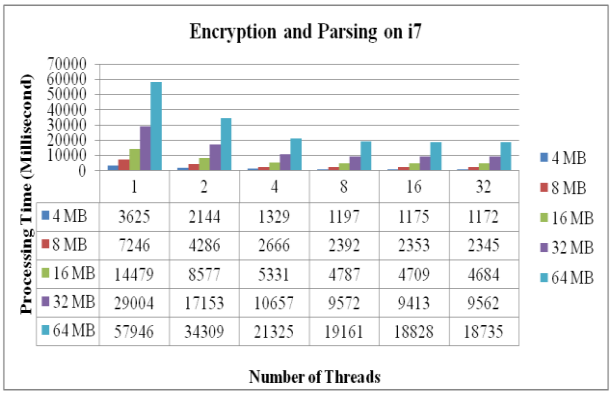
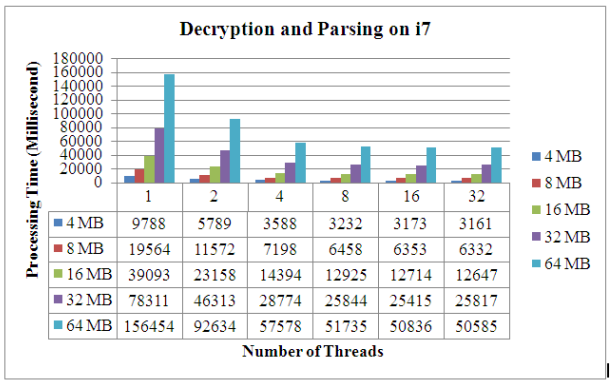


Fig.6.Parallel processing of encryption with validating parser on i7

DOI 10.17148/IJIREEICE.2016.4111

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 1, January 2016



	1	2	4	8	16	32
4 MB	9788	5789	3588	3232	3173	3161
8 MB	19564	11572	7198	6458	6353	6332
16 MB	39093	23158	14394	12925	12714	12647
32 MB	78311	46313	28774	25844	25415	25817
64 MB	156454	92634	57578	51735	50836	50585

Fig.7. Parallel processing of decryption with validating parser on i7

The similarly processing time is observed on single core, dual core and i5. Speedup i.e. time taken by program to execute is calculated using the following formula to assess the performance of the parallel model:

Speedup = Time taken by process to execute in serial (one processor)

Time taken by process to execute in parallel on j processors

Comparative chart of speedup on different cores is shown in Fig. 8. On single core system, there is slight change in processing time with increasing number of threads, when we divide SOAP message in aforesaid parts and run them in parallel. This is because single core can perform single CPU bound task at a time. So even after increasing the number of threads for parallel processing, only one thread at a time is processed on single core as there is no intervention of I/O bound task.

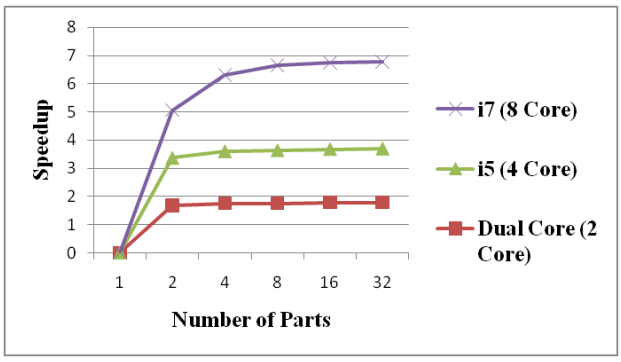


Fig.8. Comparative chart (Stacked) of speedup on different Cores

Dual core system works fine with two threads as there are two physical cores. Running two threads in parallel gives speedup of 1.6x and utilizes CPU at 98%. Increasing two more threads i.e 4 threads, increases speedup as 1.76x. However, after increasing number of threads as 8, 16 and 32, the speedup increases slightly up to 1.79x.

I5 has 2 physical cores and 2 logical cores. Running two threads in parallel increases speedup by 1.69x and with four parallel threads it increases up to 1.84x. After increasing threads up to 32, it reaches up to 1.9x. In i7, there are 4 physical cores and 4 logical cores. Running two threads in parallel increases speedup by 1.69x and with four parallel threads it highly increases up to 2.71x. Speedup reaches to 3.02x with 8 threads. After increasing threads up to 32 there is not much improvement. Maximum speed up 3.09x, is observed with 32 threads running in parallel on i7.

It is observed that if message is partitioned in multiple of number of cores available in the hardware, full strength of data parallelism can be extracted with full CPU utilization. When a single part is run on i7 (4 physical and 4 logical cores), CPU utilization is 10-15%. Increasing the number of parts (4 to 32) with number of threads running, equally, it increases CPU utilization up to 98%.

It is observed that the size of SOAP message affects the time for Encryption/decryption processing. If we divide the file into parts, and run on different cores it improves the performance (as the processing time required is reduced), but performance is limited by number of cores available. Therefore, it is clear from results that to get the best performance, the number of partitions should be equal to the number of cores available in the hardware.
Further parallel processing of partitions increases the overall performance significantly.

IV. CONCLUSION

Security processing on SOAP message is a time consuming process. The methodology presented in this paper decreases the time required for SOAP message security processing. This is due to the combination of stream based security processing approach and parallelization. It is concluded that the performance of the model increases up to optimum level when work load is uniformly divided among the available cores. Further, it is concluded that the best performance is obtained when the number of partitions is equal to the number of cores resulting in full CPU utilization. This is found only in case, if the process is totally CPU bound and there is no intervention of I/O bound task. The new model has speedup the overall processing time up to 3.09x times using 4 physical and 4 logical core as compared to serial processing model. In this way the new parallel stream based processing model for WS-Security has shown significant improvement in performance over the existing serial model.
The results are extremely useful for large XML datasets and scientific computing that require more CPU cycles and processing time. Performance improvement is important to get higher system throughput and smooth availability of services. This work is also useful for resource sensitive service environments like mobile web services where processors have limited powers. Further research work in the area of web services security can include issues like parallel stream based signature processing and security processing of composite web services.

REFERENCES

[1] N. Gruschka and N. Luttenberger, “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.

[2] V. 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, Elsevier, pp. 31-39, 2014.
[3] 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.
[4] N. Gruschka, M. Jensen, L. Lo Iacono, and N. Luttenberger, "Server-Side Streaming Processing of WS-Security," In IEEE Transactions On Services Computing, Vol. 4, No. 4, 2011.
[5] N. Gruschka, N. Luttenberger, and R. Herkenhöner, "Event-based SOAP Message Validation for WS-SecurityPolicy-Enriched Web Services", Communication Systems Research Group, Department for Computer Science, Christian-Albrechts-University in Kiel, Germany,http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.9933&rep=rep1&type=pdf
[6] N. Gruschka, M. Jensen, and L. Iacono, "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, 2010, pp. 410- 415.
[7] 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.
[8] 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, 2011, pp. 7 -13.
[9] M. Priyadharshini, I. Suganya, N. Saravanan "A Security Gateway for Message exchange in Services by Streaming and Validation," In International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 3, 2013.
[10] S. Makino, K. Tamura, T. Imamura, and Y. Nakamura. "Implementation and performance of WS-Security." IBM Research Report, Tokyo Research Laboratory, 2007.
[11] H. Liu, S. Pallikara, and F. Geoffrey, "Performance of web service security," In Proceedings of 13th Annual Mardi Gras Conference, 2005.
[12] S. Makino, K. Tamura, T. Imamura, and Y. Nakamura, "Implementation and performance of WS-Security," In Int. J. Web Service Res., 1(1), pp. 58–72, 2004.
[13] Y. Wu and Q. Zhang, "A Hybrid Parallel Processing for XML Parsing and Schema Validation," In The Markup IEEE International Conference on Web Services. 2008.
[14] N. Arora, S.Kolhe, S. Tanwani, "A new algorithm for parallel stream based processing of secured SOAP message". (communicated)
[15] Official SAX Homepage, http://www.saxproject.org/
[16] The ApacheTMXML Project: Xerces2 Java Parser, http://xerces.apache.org/xerces2-j/
[17] M. A. Rahaman, M. Rits and A. Schaad, "An Inline Approach for Secure SOAP Requests and Early Validation," In Proceeding of the Open Web Application Security Project Europe Conference (OWASP), Leuven, Belgium, 2006.
[18] M. A. Rahaman, M. Rits and A. Schaad, "Towards Secure SOAP Message Exchange in a SOA", In Proceeding of the ACM Workshop on Secure Web Services (SWS), Fairfax, 2007, VA, USA.
[19] M. A. Rahaman and A. Schaad, "SOAP-Based Secure Conversation and Collaboration", In Proceeding of International Conference on Web Services (ICWS), Los Angeles, CA, USA, 2009.