PEER- REVIEWED INTERNATIONAL JOURNAL

Aarhat Multidisciplinary International Education Research Journal (AMIERJ)

ISSN 2278-5655

Impact Factor :0.948

Bi-Monthly

VOL - II | ISSUES - V | [2013-14]

Chief-Editor:

Ubale Amol Baban

[Editorial/Head Office: 108, Gokuldham Society, Dr.Ambedkar chowk, Near TV Towar,Badlapur, MS

STREAM-BASED WEB SERVICE SECURITY PROCESSING SYSTEM

Vaishali Mangalkar [#],

[#]*P.G Scholar, Department Of Information Technology,*

Sinhgad College Of Engineering, Pune University

Prof. R. S. Sonar[*],

[#]*P.G Scholar, Department Of Computer Technology,*

Smt Kashibai Navale College Of Engineering, Pune University

Pune-41, India

Rajesh Yemul[#]

[*]*Assistant Professor,*

Department Of Information Technology, Sinhgad College Of Engineering, Pune University

Pune-41, India

Abstract—

Web Services, like any other software applications, are subject to traditional security risks and issues such as authentication, authorization, encryption, secure sockets, and so on. A profound weakness of Web Services is their vulnerability to Denial of Service attacks exploiting XML processing.

Efforts have been made to secure Web Services's availability as they have become more and more popular for inter-enterprise communications. Existing security standards such as e.g. WS-Security only address message integrity and confidentiality, also user authentication and authorization. In this paper we present a system for protecting Web Services from Denial-of-Service (DoS) attacks.

In this paper we present a solution a stream-based WS-Security processing system, which enables an efficient processing in service computing and increases the robustness against different types of Denial-of-Service (DoS) attacks. The introduced system is capable of processing all standard-conforming applications of WS-Security in a streaming manner. The system can handle e.g. any order, number and nesting degree of encryption and signature operations, thus closing the gap towards more efficient and dependable Web Services.

INTRODUCTION

Web services implement a new paradigm of distributed communication called service oriented architecture (soa).the recommended protocol for communication is soap, which is xml based and message oriented.

Ws-security is a standard providing message-level security in web services. it ensures integrity, confidentiality and authenticity. when using a standard dom approach for xml processing, the web services servers easily become a target of denial-of-service attack. soap-based web services enable flexible software system integration especially in heterogeneous environments, and is a driving technology for inter-organization business process. since soap is an xml-based protocol, it inherits lot of advantages of text-based xml such as message extensibility, human readability and utilization of standard xml processing components.

This paper presents how a secured soap message as defined in ws-security can be completely processed in streaming manner. it can handle e.g. any order, number and nesting degree of signature and encryption operation. Thus proposed system also provides increased robustness against denial-of-service (dos) attacks.

EXISTING SYSTEM

The main problems-used by critics since the start of web services-are verbosity of transmitted messages and high resource requirements for processing. Most issues are further increased when using soap security through the need of handling larger messages and performing some cryptographic operations. these issues possess performance challenges which need to be addressed and solved to obtain the efficiency and scalability required by large information systems. these issues become severe e.g. in mobile environments with limited computing resources and low data rate network connections, or for high-volume web service transactions comprising a large number of service invocations per second.

BACKGROUND AND MOTIVATION

Some motivation foundations are introduced in the following sections.

A. Efficient Processing.

xml document is efficiently processed using stream-based processing in which xml document is read and parsed step by step and passed to the application. there exist two general processing models for xml documents: document-based processing (dom model) and stream-based processing

with dom model approach, the complete xml document (e.g. a web service soap message) is read, parsed and transformed into an in-memory object tree representation of the document. complete xml processing is thus performed using this object tree.dom trees can be navigated freely and parsed arbitrarily, and also provides maximum flexibility for developers. streaming refers to a programming model in which, the xml document is read and parsed step by step, divided into parts (e.g. single xml elements) and passed to the application, which then operates on these xml parts. two approaches can be distinguished for implementing the communication between the application and parser.

- **Pull Parsing Vs Push Parsing**

with pull approach client application calls methods on an xml parsing library when it needs to interact with an xml infoset--that is, the client only gets (pulls) xml data when it explicitly asks for it.

With push approach an xml parser sends (pushes) xml data to the client as the parser encounters elements in an xml infoset--that is, the parser sends the data to the client even if it is not ready to use it at that time.

Advantages of pull parsing over push parsing when working with xml streams:

- With pull parsing approach, the client as per the need can call method methods on the parser and also controls the application thread. whereas, with push processing approach, parser plays a vital role in controlling the application thread, and the client accept invocations from the parser.

- Pull parsing libraries are much smaller and the client codes to interact with those libraries much simpler than with push libraries, even for more complex documents.
- Pull clients can read multiple documents simultaneously.
- Xml documents are filtered by a stax pull parser in such a way that elements unnecessary to the client can be ignored, and it can support xml views of non-xml data.

B. Dos Attacks And Robustness:

Denial of service (dos) attacks aim at reducing or completely eliminating a system's or service's availability. various two kinds of dos attacks can be distinguished: protocol deviation attacks and resource exhaustion. protocol deviation attacks exploit vulnerabilities in implementations of protocol processing entities. in some cases a single packet that diverges from the intended protocol flow can make the attacked system crash. a well-known example is ping of death. resource exhaustion attacks consumes the resources necessary to provide the service (network bandwidth, memory and computation resources). using such an attack makes it difficult to completely interrupt a service's availability. one classic way to perform resource exhaustion attack is to query a service using a very large request message. this is called as oversize payload attack coercive parsing attack use a large number of namespace declarations, oversized prefix names or namespace uris or very deeply nested xml structures.thus,stream-based processing not only increases efficiency but also-as an outcome the robustness against resource exhaustion dos attacks such as coercive parsing and oversize payload[1]

C. Immediate Fault Detection

one of the most important aspects of stream-based processing approach is it detects the bogus messages more timely the tree based approach parses and processes whole xml document before any application-specific operation can be initiated and transforms it into an in memory representation of the xml element's tree structure. the flaws of tree based approach become very clear in the figure 1 (a) where xml document must be read completely before any processing on the contained xml elements can start. it thus enables a malicious soap message sender to feed in a huge xml document of schema-violating contents, which the parser must read in completely

before being able to detect the presence of a schema violation. in such a way, sender can control the size of the soap message wherein, an attacker can cause heavy workloads for parsing xml documents at the server [2], [3]. the stream based xml processing model starts to parse and process contents of the document block wise, e.g. reading 15 kb of data from the xml file, and then a stream-based parser is thus run on this fragment only before processing the next 15 kb.such scene is depicted in figure 1(b).

Stream-based approach enables an early processing of xml contents for the special case of schema violation enabling server application to cut off the connection. stream-based processing thus helps one to access identity, authentication and authorization information contained in the request much earlier. in terms of performance stream-based processing approach turns out to be superior-compared to tree-based approach, but when an xml document contains encrypted data fragments the performance advantage vanishes the most convenient approach consists of storing the encrypted block completely in memory, then applying the cryptographic operation of decryption to it—resulting in a new, plain xml fragment. this new xml fragment is parsed and processed subsequently. this approach is shown in figure 1 (c).

But this leads to the same issues with regard to attack exploitability. Here total memory consumption rise, as the encrypted contents have to be stored twice (cipher text and plain text).an attacker could send schema-invalid contents inside a huge encrypted block of soap message. thus, a more efficient approach consists of decrypting encrypted xml document parts also parse and process them immediately. this approach is illustrated in figure 1 (d). in terms of performance stream-based processing approach turns out to be superior compared to tree-based approach, but when an xml document contains encrypted data fragment. the most convenient approach consists of storing the encrypted block. in case of any schema invalidity or any other xml processing fault within the encrypted contents, stream-based approach provides highly efficient way to determine the fault as soon as possible. the reception of the full message can be cut off, thereby reducing the performance overhead of processing

D .Open Challenges With Defined Solution

high resource consumption and the ability to detect malicious messages make security processing an ideal candidate for streaming methods. also, it requires certain complex operations on the soap message. thus, a comprehensive and fully-integrated stream-based ws-security processing engine is required to effectively use stream-based soap processing. according to the

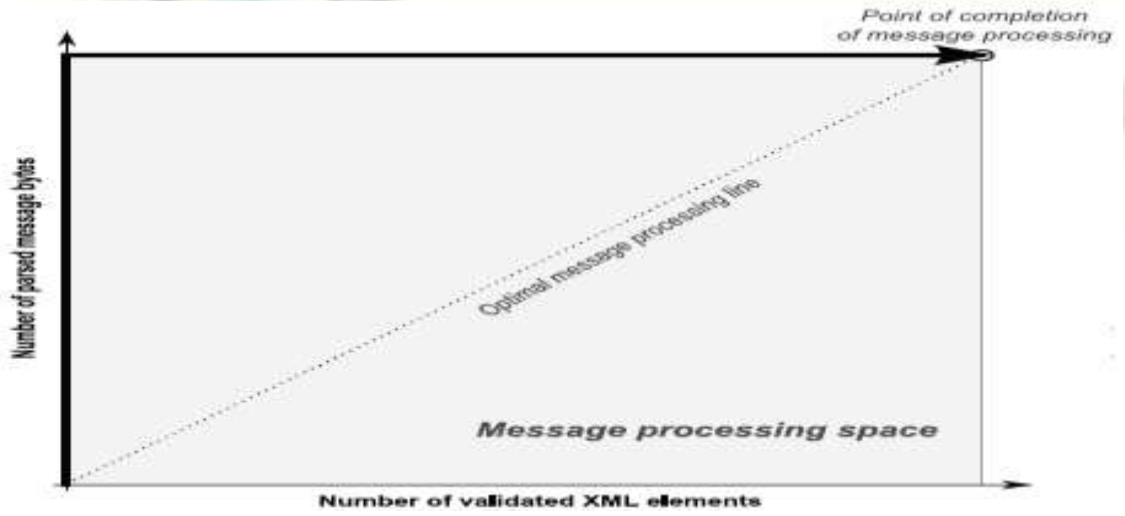
study, to date there exists no comprehensive stream-based ws-security engine defined in ws-security.

there are some shortcomings on stream-based processing of xml signatures and stream-based encryption and decryption of xml documents, which is not sufficient for full ws-security processing integrated into overall soap processing.

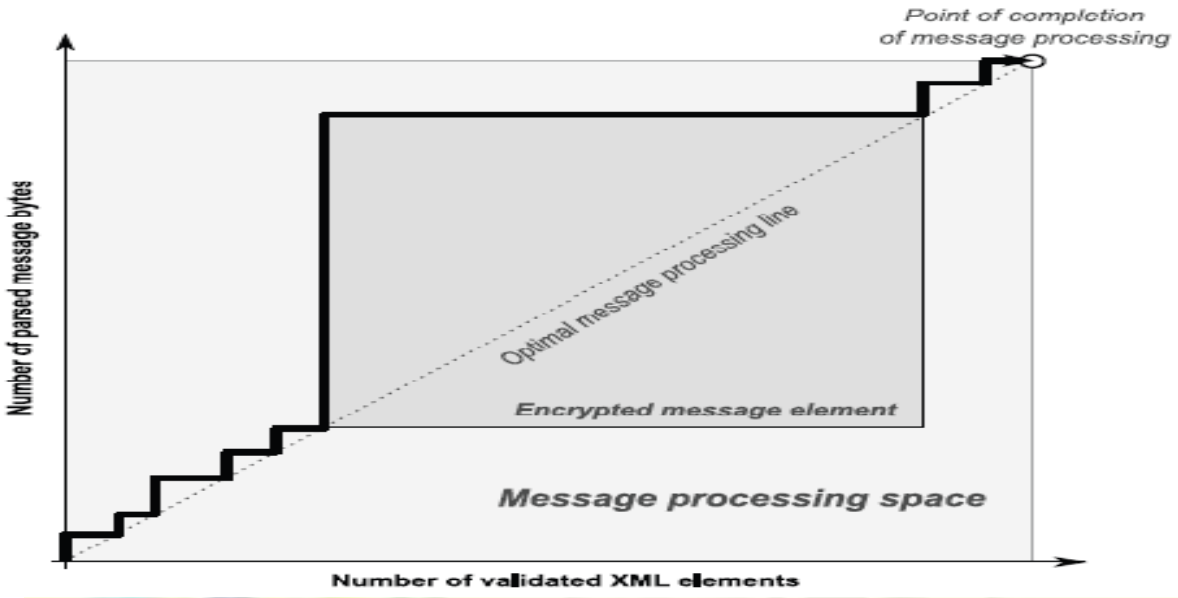
in this paper, the concepts and algorithms for a comprehensive stream-based ws-security engine are introduced and discussed. the developed engine will thus have the following capabilities.

- xml signatures can be processed with backward references
- handling combination of signature and encryption in any order, number and nesting degree of resolution of cryptographic material including encrypted keys
- amalgamation with streaming ws-securitypolicy evaluation including streaming xpath evaluation
- inclusion with streaming access control decision.

thus, the system presented provides the missing link to a fully streamed soap processing which allows to leverage the performance gains of streaming processing as well as to implement services with an increased robustness against denial-of-service (dos) attacks



(a)tree-based (dom) xml processing scheme



(c)semi stream-based xml processing with encrypted part

fig. 1: graphical comparison of different xml processing schemes

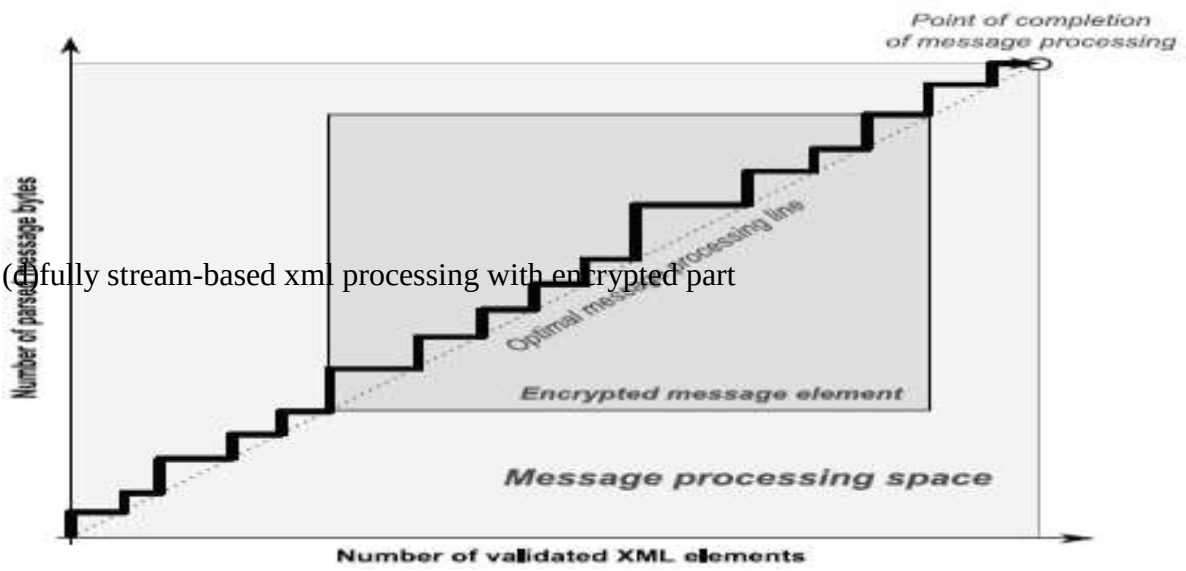
IV. STREAMING WS-SECURITY PROCESSING

in this section, the algorithms for processing ws-security enriched soap messages in a streaming manner are presented and discussed

A. Ws-Security

the most important specification addressing the security needs of web services is ws-security [4]. it collaborates with the soap specifications, providing integrity, confidentiality and authentication for web services. ws-security defines a soap header block called security header- that carries the ws-security extensions. additionally, it defines how existing xml security standards like

(b)stream-based (sax) xml processing scheme



(c)fully stream-based xml processing with encrypted part

xml signature and xml encryption [5] are applied to soap messages. however, the use of security mechanisms in soap messages supports attacks on web services availability.

following steps must be performed (order not necessary) for processing a ws-security enriched soap message at the server side.

- ws-security header processing.
- verification of signed blocks
- decryption of encrypted blocks

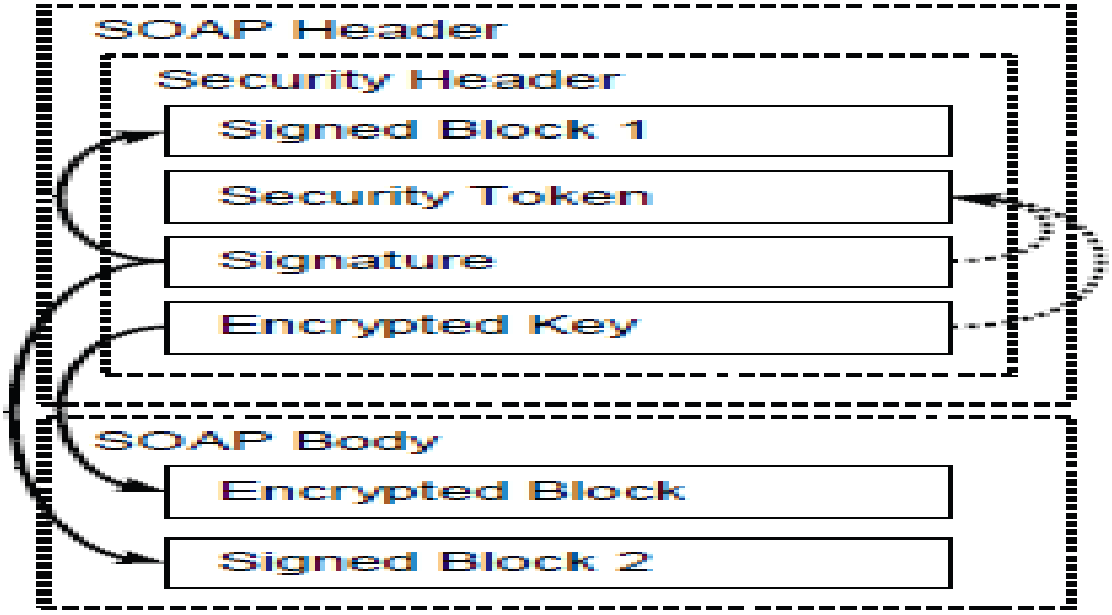


fig 2 ws-secured soap message

B. Ws-Security Secured Soap Message:

figure 2 shows an example of a ws-security-secured soap message containing references. security tokens contain identity information and cryptographic material (typically an x.509 certificate) and are used inside signatures and encrypted keys and are backward referenced from those. encrypted key elements contain a (symmetric) cryptographic key, which is asymmetrically encrypted using the public key of the recipient. this symmetric key proves to be useful for encrypting message parts (at the client side) and also for decrypting the encrypted blocks (at the server-side). encrypted keys must occur inside the message before the corresponding encrypted blocks.

C. Architecture

the figure 3 shows the architecture of the system for stream based processing of ws-security enriched soap messages called checkway. it operates on sax events created by a sax parser and contains four types of event handlers

the first handler is responsible for processing the ws-security header which has a fixed defined position inside the soap message, so the handler can be statically inserted inside the handler processing chain as the headers. however this goes different for signed and encrypted blocks wherein, they may occur at nearly arbitrary positions inside the soap message, and also can be nested inside each other.

thus, the dispatcher handler is responsible for detecting signed and encrypted blocks and inserting a respective handler into the processing chain (at which position will be discussed below).while detecting encrypted blocks is trivial (they start with the element xenc:encrypteddata), as signed blocks are not explicitly marked so detecting them becomes more difficult. in the case of forward references, the signature elements are (regarding the document order) before the signed block. therefore, it becomes easy to detect forward referenced signed blocks by comparing the id attribute of that element with the list of references from the signature elements processed before. whereas, if backward references are considered there is no possibility for a definite decision if an element is signed or not signed. the following solution for this problem

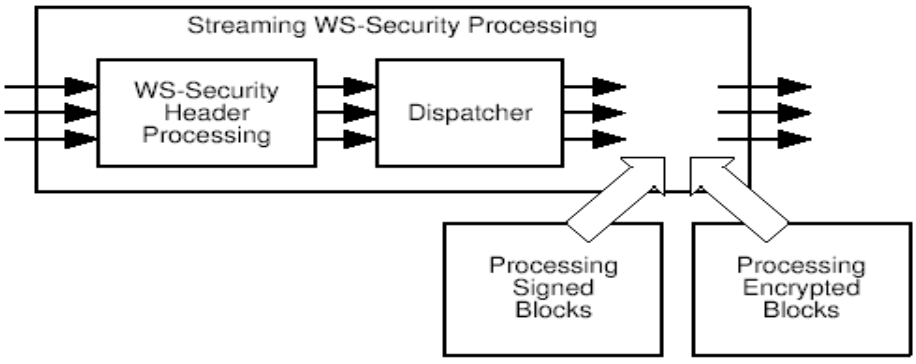


fig 3 architecture of a stream-based ws-security processing system

has been developed. every element before the end of soap header (only there backward references are possible) that contains an id attribute is regarded as potentially signed and therefore the "signed block processing" is started. the id and the result of the signed block processing (i.e. the digest of this block) is stored at the end of such block. when processing a signature the included references are compared to the ids stored from the potentially signed blocks and the stored digest is verified by comparison to the one inside the signature element.

D. Ws-Security Header Processing Automaton

figure 4 shows the automaton for the event-based processing of the ws-security header. the implementation is started by the element wsse:security inside the soap header. this element can occur more than once inside a soap message, but only once per recipient. as one can see in the automaton, the ws-security header can contain security tokens, timestamps, signatures, encrypted keys and references to encrypted blocks in arbitrary order. following methods are invoked during processing of the ws-security header :

- storetoken(ref, char): the value char is decoded to the key key and stored inside the list key (2)
- checktimestamp(c, e): check, if the timestamp is valid, e.g. if e > c and e < now (with now the current time) (3)
- storereference(ref, type): add (ref, type) to the end of the list ref (7)

E. Encrypted Key Processing Automaton

figure 5 shows the automaton for processing an xenc:encryptedkey element contained in the wssecurity header. the processing starts with reading the encryption algorithm alg (1).inside the ds:keyinfo element (2) a hint to the key pair key_{priv} and key_{pub} is given .

the key key_{priv} is used for initializing the decryption algorithm inside the function in it decryption(alg) (4). the function decrypt(char) decrypts then the content of the xenc:cipherdata element using this algorithm in conjunction with key_{priv}. the result is the (symmetric) key, that is used later to decrypt encrypted content.

the references stated inside the xenc:referencelist claim the usage of the current key for those encrypted blocks. thus the storekey(...) function adds the pair (ref, key) to enckey (7) to enable the decryption of the appropriate encrypted block. the pair(ref,enc)is added additionally to the end of the list of security ref.

F. Key Information From A Ds: Keyinfo Element

the xml signature specification defines a <ds:keyinfo> element which is used to provide information about the "key" used in the signature. for token references within signatures, it is recommended that the <wsse:securitytokenreference> be placed within the <ds:keyinfo>.

this element then contains a reference to a security token, which may be stored either external or inside the same message. the token can occur either before the token or as a direct child element, if the security token is inside the same message. in either case, at the moment the ds: keyinfo is read; the related security token is already known or can at least be effectively obtained.

the cryptographic keys needed for decryption and signature verification can be derived from the security token. different possibilities are used for this procedure. common case is the use of an x.509 certificate as security token which contains the public key for signature verification. the associated private key is locally stored at the server system and can be identified through the identifiers inside the certificate.

G. Signature Processing Automation

figure 6 shows the automaton for processing <ds: signature> element from the ws-security header. in order to verify the signature value, the <ds: signedinfo> block must be canonicalized and hashed. thus, at the begin of that element the canonicalization and hashing is started by the function starthashing (1). the hashing algorithms for the ds: signedinfo block are read (2).

H. Encrypted Block Processing Automation

while processing a message, if an element xenc: encrypteddata (with id ref) is read, then the dispatcher creates a new instance of the encrypted block handler and inserts it into the

handler chain as mentioned above. the encrypted block handler itself implements the event processing shown in the automaton in figure 5. it begins with reading the encryption algorithm alg (1). there are two possibilities for the key needed for decryption. either it is given inside the ds: keyinfo element (2) or it is given by an encrypted key

I. Signed Block Detection And Processing

signed blocks can be detected by two methods. the first method is the one in which all elements containing an id attribute must be considered as potentially signed before the ws-security header has been processed. for those elements a new instance of the signed block handler is created and inserted into the event chain.

in the second method the signed block handler is removed from the event chain and opendigest is checked at the end of the soap message. if it is not empty, a block referenced from a signature is missing and it is understood that the signature is invalid. the soap message processing is stopped with an exception reading the encryption algorithm alg (1).a hint is provided to the key pair key_{priv} and key_{pub} . the key key_{priv} is used for initializing the decryption algorithm inside the function

CONCLUSION

it is a well-accepted truth today that event-based xml processing approaches like sax are superior to tree-based processing when it comes to performance issues. whereas on the other hand, the event-based approach requires a big conceptual change in the way applications are designed.

as a result we have introduced a comprehensive framework for event-based ws-security processing system, which enables a more efficient processing and increases robustness against various types of dos attacks. the main advantages of the streaming model include an increased efficiency in terms of resource-consumption and an enhanced robustness against different kinds of dos attacks.

thus the proposed solution can handle any order, number and nesting degree of signature and encryption operations filling the gap in a stream-based processing chain towards more efficient and dependable web services.

REFERENCES

N. Gruschka And N. Luttenberger, "Protecting Web Services From Dos Attacks By Soap Message Validation," In Proceedings Of The Ifip Tc-11 21. International Information Security Conference (Sec 2006), 2006, Pp. 171–182.

N. Gruschka, M. Jensen, And N. Luttenberger, "A Stateful Web Service Firewall For Bpel," In Proceedings Of The 2007 Ieee International Conference On Web Services (Icws 2007), 2007, Pp. 142–149.

M. Jensen, N. Gruschka, And N. Luttenberger, "The Impact Of

Flooding Attacks On Network-Based Services," In Proceedings Of The Third International Conference On Availability, Reliability And Security(Ares 2008), 2008, Pp. 509–513.

W. Lu, K. Chiu, A. Slominski, And D. Gannon, "A Streaming

Validation Model For Soap Digital Signature," In The 14th Ieee

International Symposium On High Performance Distributed Computing (Hpdc-14), 2005.

T. Imamura, B. Dillaway, And E. Simon, "Xml Encryption Syntax

And Processing," W3c Recommendation, 2002.

"Pdca12-70 Data Sheet," Opto Speed Sa, Mezzovico, Switzerland.

A. Karnik, "Performance Of Tcp Congestion Control With Rate Feedback: Tcp/Abr And Rate Adaptive Tcp/Ip," M. Eng. Thesis, Indian Institute Of Science, Bangalore, India, Jan. 1999.

J. Padhye, V. Firoiu, And D. Towsley, "A Stochastic Model Of Tcp Reno Congestion Avoidance And Control," Univ. Of Massachusetts, Amherst, Ma, Cmpsci Tech. Rep. 99-02, 1999.

Wireless Lan Medium Access Control (Mac) And Physical Layer (Phy) Specification, Ieee Std. 802.11, 1997.