

A Security Framework To Prevent Xml Attacks And Provide DoS In Service Oriented Architecture (SOA)

¹A.Murugan, ²Dr.K.Vivekanandan, ³R.Parthasarathy

¹Asst.Prof. Department of Computer Science & Engineering, SRM university

*²Professor, Department of Computer Science & Engg,
Pondichery Engineering, College*

*³R.Parthasarathi, Research Graduate (Enigeering Management),
University of Alberta, Canada.*

E-mail : ¹ murugan.abap@gmail.com, ² k.vivekanandan@pec.edu

ABSTRACT

The on-line transactions that are happening in now a day's takes place based on the web service. Therefore to provide appropriate security to prevent the web service from any vulnerability becomes most important. Although there are several attacks that are happening in today, this paper focuses on SOAP related XML attacks and DoS attacks. XML is universally accepted SOAP format, which is the basic for communication in Service Oriented Architecture (SOA). XML attacks could happen if an attacker gains access over the underlying network between client and server. The attacker alters the message to perform desired action. E.g.:- Modifying the Account number or Amount in a bank transaction. In existing systems, only selected parts of the xml are chosen for encryption. XML Schema hardening by XML Signature wrapping using Symmetric key encryption requires a trusted service to be maintained for key management. To overcome this, a MD5 hash of the message is generated at the client and sent along with the message. This hash value will be used to check the integrity of the individual messages. Even this mechanism can be cracked, if the attacker performs traffic analysis and identifies the mechanism used. To overcome this, the behavior of MD5 algorithm should be changing dynamically. This is achieved by initializing the MD5 algorithm's buffer with values that are unique at a given point of time i.e. for different time the buffer values will not be same. Another form of attack is a flooding attack where those attackers choose particular web service and flood them with n number of requests, so that the Web Service would stall and stop processing further requests. Detection of DoS attacks is possible by performing traffic analysis on the rate at the requests arrive and detect any

flash crowds. To provide DoS, we maintain the IP address of each request in the server and deny the service to the client if the number of requests from a single client reaches above the threshold value.

Keywords: DDos distributed Denial of service, UDDI universal data discovery and integration, WSDL web service description language

1. INTRODUCTION

Web Services and *Service-Oriented Architectures* (SOAs) are often considered to be among the most important technological innovations over the last few years. Nevertheless, the benefits of these new approaches stand against some serious flaws these new technologies bring along. The most severe issues include *Web Service security*.

The typical requirements for a secure system are *integrity, confidentiality and availability*. Any action targeting at violation of one of these properties is called an attack, the possibility for an attack is called vulnerability.

In a *Service-Oriented Architecture* environment, the interaction between the client and the server would be mostly in the form of XML message, since it is most universally acceptable.

This article discusses about the list of security issues in the domain of Web Services and provides counter measures to prevent *XML attacks* as referred in Fig 1.1 and *Denial-of-Service*(DoS) referred in refer Fig 1.2.

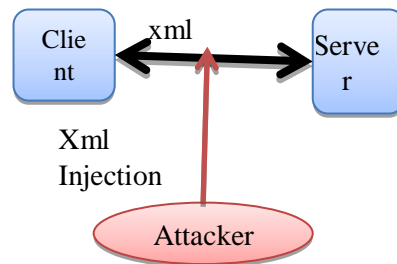


Fig 1.1 Example of XML attacks

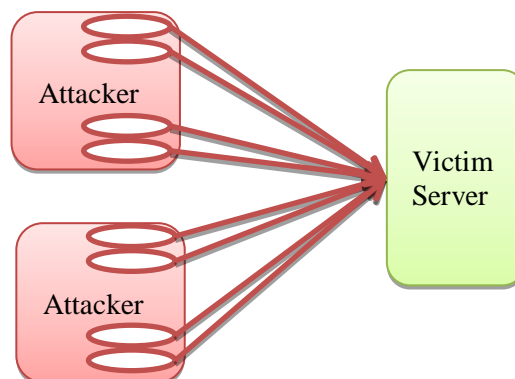


Fig1.2 Example of DoS

WEB SERVICES ATTACKS

Brute force attacks - Attacks that use the raw computer processing power to try different permutations of any variable that could expose a security hole.

Buffer overflows –The maximum size of a given variable (string or otherwise) is exceeded, forcing unintended program processing. In this case, the attacker uses this behavior to cause insertion and execution of code in such a way that the attacker gains control of the program in which the buffer overflow occurs. Depending on the program's privileges, the seriousness of the security breach will vary. Often, the unintended methods of access can be less secure deprecated methods.

Cookie manipulation -Through various methods, an attacker will alter the cookies stored in the browser. Attackers will then use the cookie to fraudulently authenticate themselves to a service or Web site.

Cookie replay attacks- Reusing a previously valid cookie to deceive the server into believing that a previously authenticated session is still in progress and valid.

Cross-site scripting- An attacker is able to inject executable code (script) into a stream of data that will be rendered in a browser. The code will be executed in the context of the user's current session and will gain privileges to the site and information that it would not otherwise have.

Data tampering-An attacker violates the integrity of data by modifying it in local memory, in a data-store, or on the network. Modification of this data could provide the attacker with access to a service through a number of the different. Denial of service (DoS)-It is the process of making a system or application unavailable. For example, a DoS attack might be accomplished by bombarding a server with requests to consume all available system resources, or by passing the server malformed input data that can crash an application process. Dictionary attack Use of a list of likely access methods (usernames, passwords, coding methods) to try and gain access to a system. This approach is more focused and intelligent than the “brute force” attack method, so as to increase the likelihood of success in a shorter amount of time. Session hijacking also known as man-in-the-middle attacks, session hijacking deceives a server or a client into accepting the upstream host as the actual legitimate host. Instead, the upstream host is an attacker's host that is **manipulating** the network so the attacker's host appears to be the desired destination.SQL injection - Failure to validate input in cases where the input is used to construct a SQL statement or will modify the construction of a SQL statement in some way. If the attacker can influence the creation of a SQL statement, he or she can gain access to the database with privileges otherwise unavailable and use this in order to steal or modify information or destroy data

2. LITERATURE WORK

Service Oriented Architecture is an architectural paradigm and discipline that may be used to build infrastructures enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate domains of technology and ownership. Besides SOAP and UDDI, which make the foundation of SOA; WSDL also plays an important role in this architecture. So far, in most of the security solutions that have been offered for SOA, providing security of SOAP messages has been the main objective. But in this article, the security view has been changed to WSDL files. So a new framework has been proposed which aims to protect Web services against WSDL attacks. Additionally to the best of our knowledge at the time of the writing of this article no other practical solution has been suggested in order to secure Web services WSDL files in SOA environment. Also, in order to provide security requirements, a new extension of WSDL file in the suggested framework has been offered as [4].

In the context of security of Web Services, the XML Signature Wrapping attack technique has lately received increasing attention. Following a broad range of real-world exploits, general interest in applicable countermeasures rises. However, few approaches for countering these attacks have been investigated closely enough to make any claims about their effectiveness. The effectiveness of the specific countermeasure of XML Schema validation in terms of fending Signature Wrapping attacks. To investigate the problems of XML Schema validation for Web Services messages, the approach of Schema Hardening, a technique for strengthening XML Schema declarations [3].

Business Intelligence or e-commerce applications are increasingly built on the Web Service platform. Thus, SOAP-related attacks have a higher chance of occurring at the Application Layer [2]. Although active research has been on-going in Host and Network-based intrusion detection and intrusion prevention areas, they are not adequate to countermeasure the attacks occurring at the Application Layer. This is detrimental, especially for e-commerce where sensitive and huge amount of business-related information are being exposed over the Internet. Consequently, in this paper, a policy-enhanced fuzzy model with adaptive neuron-fuzzy inference system features is introduced. Transactions generated by simulation reveal that SOAP-related attacks at the Application Layer can be detected and prevented by validating input values, input field lengths, and SOAP size using our model to classify the possibilities of granting or denying access to the backend application or database. Restricting the inputs using business policies further strengthens the model to be able to achieve detection accuracy of 99% and false positive rate of only 1%. Thus, our model has significantly contributed to an added layer of security protection for Web Service-based e-commerce applications. Service-Oriented Architecture (SOA) is an architectural style whose primary goal is to achieve minimal dependency among interacting software agents. And as with all new technologies, it comes with its share of challenges. Of particular difficulty is the challenge of securing a service oriented system. Since Web services supply a significant way to provide SOA requirements, any brought up issues, like security of SOA, can be related to Web services. On the other hand, Web services are well-known XML1_based technologies [5]. So the security of Web

services can be directly affected by XML security. In this study, a new security framework is proposed which aims to defend main XML threats, especially WSDL attacks in an SOA environment. To the best of our knowledge, it is for the first time that such a practical solution has been offered, which not only handle one aspect of XML vulnerabilities like SOAP2 messages but, also try to defend WSDL3 threats, in an SOA environment.

Distributed denial-of-service (DDoS) attacks are considered to be among the most crucial security challenges in current networks because they significantly disrupt the availability of a service by consuming extreme amount of resource and/or by creating link congestions. One type of countermeasure against DDoS attacks is a filter-based approach where filter based intermediate routers within the network coordinates with each other to filter undesired flows. The key to success for this approach is effective filter propagation and management techniques. However, existing filter-based approaches do not consider effective filter propagation and management. [1] Define three necessary properties for a viable DDoS solution: how to practically propagate filters, how to place filters to effective filter routers, and how to manage filters to maximize the efficacy of the defense. We propose a novel mechanism, called Adaptive Probabilistic Filter Scheduling (APFS) that effectively defends against DDoS attacks and also satisfies the three necessary properties. In APFS, a filter router adaptively calculates its own marking probability based on three factors: 1) hop count from a sender, 2) the filter router's resource availability, and 3) the filter router's link degree. That is, a filter router that is closer to attackers, has more available resources, or has more connections to neighbors inserts its marking with a higher probability. These three factors lead a victim to receive more markings from more effective filter routers, and thus, filters are quickly distributed to effective filter routers. Moreover, each filter router manages multiple filters using a filter scheduling policy that allows it to selectively keep the most effective filters depending on attack situations. Experimental results show that APFS has faster filter propagation and a higher attack blocking ratio than existing approaches that use fixed marking probability. In addition, APFS has a 44% higher defense effectiveness than existing filter-based approaches that do not use a filter scheduling policy.

3. PROPOSED ARCHITECTURE

The integrity of the XML message is preserved by generating a MD5 Hash value for the whole message. Even this generated hash could be cracked if the attacker is given a stipulated amount of time. To avoid this buffer values of the encrypting algorithm are initialized dynamically with the current timestamp. This way for a same input, the MD5 would produce different outputs at different given point of time. One limitation with this scheme is that the both the client and the server must be synchronized with time. Fig.3.1 shows the architecture of how xml attack is prevented.

The example below show XML encryption scheme used for encrypting the message.

$$\text{XML} + \text{MD5} + \text{Time Stamp} = \text{Hashvalue}$$

$$\text{Hashvalue} + \text{XML} = \text{Message.}$$

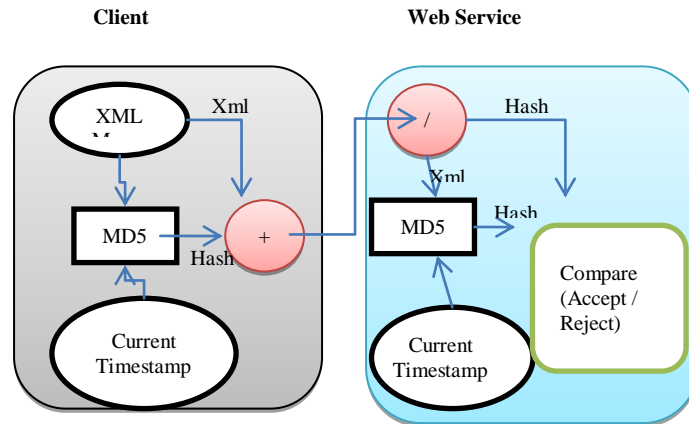


Fig 3.1 Preventing XML attacks

Denial of Service is achieved by maintaining the list of active IP addresses and their corresponding active request counts that are being processed currently. This list is maintained in the server's session. Whenever the system receives the request it makes an entry in the list. If such IP exists then the count is incremented. Appropriate action is taken when the request is processed and the system replies with the response. A threshold value is obtained based on the list and the load that the service with could withstand. Whenever a request for a particular machine is reached to a threshold value then all the requests from that particular source is rejected. Threshold value will fall down when more requests are processed. So this dynamic changing nature of the threshold value plays a vital role in providing enhanced DoS. Fig 3.2. Shows that how DoS is prevented.

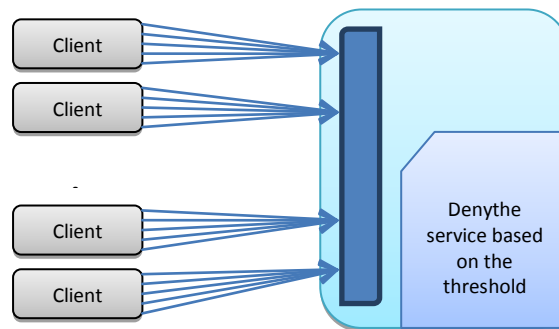


Fig 3.2. Dos Prevention

4. IMPLEMENTATION

Hash Generation and Comparison

Hash generation is done both at the client and the web service (server), whereas comparison is done only at the Web Service. Algorithm used to generate the hash is MD5. Input the hash algorithm will be the actual Xml message and the current timestamp. The buffer values in the MD5 are replaced with the values in the current timestamp. Actual Xml message is passed as input to the MD5 algorithm to generate the hash value. A schematic diagram representing the hash generation is given in fig.4.1 and hash comparison is given in fig 4.2

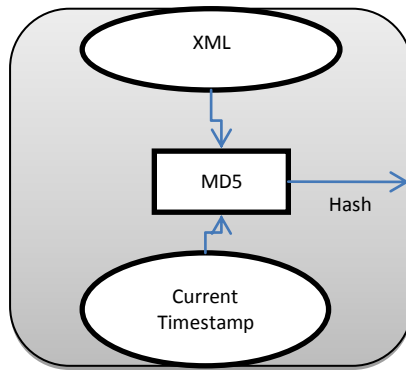


Fig 4.1. Hash Generation.

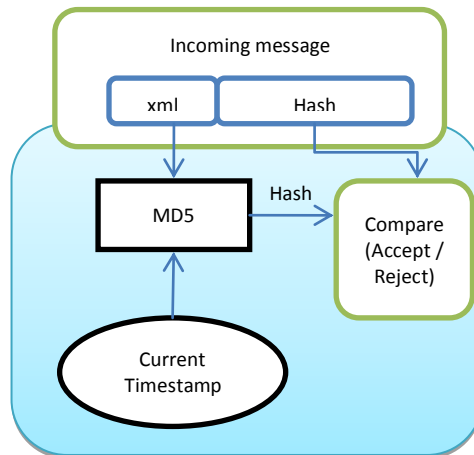


Fig 4.2 Hash Comparison

XSD Validation

XSD validation is done by reading the input xml message node by node. While reading each node, it is verified against the schema that is pre-defined. If XSD validation fails then the system will not process the requests further as shown in Fig. 4.3

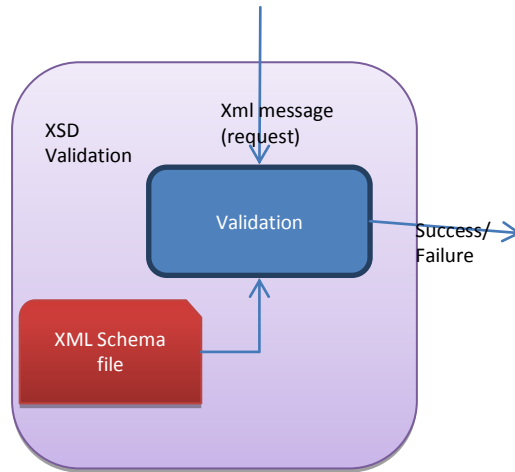


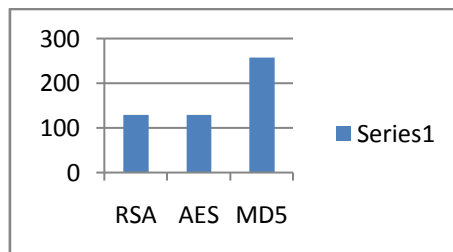
Fig4.3. XSD Validation

Dynamic threshold modification and denying the service.

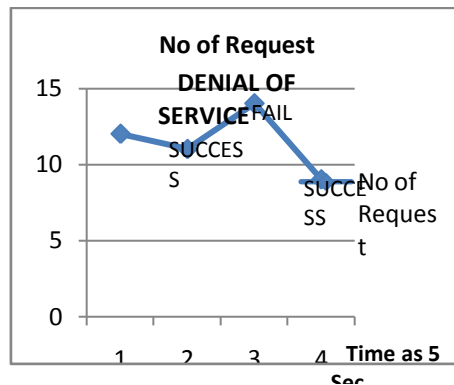
Threshold value is calculated dynamically based on the number of requests that are being processed currently and the load that the system can withstand. Whenever a request arrives which exceeds the threshold value then that particular request will be denied until further more requests are processed completely.

5. EXPERIMENTAL RESULTS

A user enters the number and the amount for which the transaction is to be carried out. Generated XML section displays the corresponding XML request message that is to be sent to the server. When “Call Web service” is used, the XML request message is sent to the web service and waits for the response. After the request is processed, the status is displayed in the Server Response section. User can send n number of requests from the single instance by selecting the Multiple Request and entering the number of request that has to be sent to the service.



A scenario where the xml message is not altered. Hence the Service has replied with the response message.



Whenever the xml message is altered the service will reply with the Failure message. Here the amount value in the xml message is altered hence the service has replied with the failure message.

Whenever the request reaches the threshold value for a particular period of time the service will deny further more requests from the service.

6. CONCLUSION

The hash generation algorithm is changed dynamically by replacing the buffer values, which makes the algorithm more difficult to crack. Even though several schemes have been implemented to enhance DoS, this scheme can define-tuned to produce better results. This can be achieved by either changing the algorithm or by using efficient. Clock synchronization is another demanding factor to prove this scheme. In future some other mechanism can be used to replace the buffer values instead of using the timestamp. Same methodology can be implemented with different hashing algorithms like SHA256, SHA512 etc. and results can be compared for better analysis.

REFERENCES :

- [1] Adrian Perrig, Dongwon Seo, Heejo Lee, "APFS: Adaptive Probabilistic Filter Scheduling against distributed denial-of-service attacks" ELSEVIER International conference on Computers and Security 2013.
- [2] Chien-Sing Lee, Gaik-Yee Chan, Swee-Huay Heng, "Policy-enhanced ANFIS model to counter SOAP-related attacks", ELSEVIER International conference on Knowledge-Based Systems 2012.
- [3] Christopher Meyer, Juraj Somorovsky, and Jorg Schwenk, Meiko Jensen "On the Effectiveness of XML Schema Validation for Countering XML Signature Wrapping Attacks" IEEE International Conference 2011.
- [4] Narges Shahgholi, Mehran Mohsenzadeh, Mir Ali Seyyedi, Saleh Hafez Qorani, "A New SOA Security Framework Defending Web services Against WSDL Attacks" IEEE International Conference 2011.

- [5] Narges Shahgholi, Mehran Mohsenzadeh, Mir Ali Seyyedi, Saleh Hafez Qorani, "A new security framework against Web services' XML attacks in SOA" IEEE International Conference 2011.