ISSN: 2321-7782 (Online)
Volume 3, Issue 6, June 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A SOA Security Solution to Prevent Replay Attacks

Mohamed Ibrahim B[1]
Software Solution Architect &
Research Scholar
Tiruchirappalli - India

Mohamed Shanavas A R[2]
Associate Professor
Jamal Mohamed College (Autonomous)
Tiruchirappalli - India

Abstract: ***In Service Oriented Architecture (SOA), the service a Software component meant to do a defined functionality. These services do not depend on the implementation of other services. Thus the services in SOA work on Loosely Coupled Architectures where one application can be altered without affecting the other applications in an enterprise. Also these services which are part of an application run independently with its own memory and computing power. A web service is the latest implementation of SOA and the preferred choice of industry to interconnect the relevant applications in enterprises. However, this web service architecture leads to several attacks as it works on request/response mechanisms; and these request/response messages are standardised to follow simple XML format. One such kind of SOA attacks is "Replay" attack, where the intruder captures the valid request message by sniffing the network packets and tries to keep on sending the captured message to the web service providing server to make it stressed and result in not responding to the requests. Even though there are some security solutions available in terms of hardware components, they will not detect all the replay attacks and there are changes to stop the legitimate request as these hardware components work on routing and counting the number of packet received during certain period of time. In this paper, we proposed a SOA security solution which prevents the replay attacks at application level. As the implemented algorithm in the proposed solution intelligently works that interprets the message headers and filters only request that are most likely to be attacked, it will not stop any valid request from other clients. As part of Proof-of-Concept (PoC), the proposed solution is developed using Java technologies and tested in a real-time SOA enterprise environment.***

Keywords: SOA; Web Services; Replay Attacks; DoS; SOAP.

I. INTRODUCTION

The Service Oriented Architecture (SOA) has emerged as an alternate paradigm to the legacy way of enterprise computing. Among the SOA implementation technologies, the web service is the choice of enterprise which works on simple Internet protocols [1]. It incorporates the SOA goals such as reuse, modularity, component-based, interoperability and granularity. Basically the SOA follows Loosely Coupled architecture, where the loosely coupled is the behaviour of systems that reduces the interdependencies across its modules [2].

In Web Services architecture, the list of services is called web methods which form a web service. A single web service or a group of web services will be running on a web server. The web method names and their input/output parameters are described using Web Services Description Language (WSDL) and published into service registry. The web service consumers who are also called as web service clients who will search the required service information from the service registry and start invoking the required services from the web server. The web service follows request/response pattern while servicing for its clients. The high-level view of SOA client-server interaction is given in Fig. 1.

As the message format used for request and response is in simple XML format, Simple Object Access Protocol (SOAP), web services are prone for attacks. One such kind of attacks is replay attack [3], in which the attacker captures a legitimate

SOAP request and uses this request to attack the web server by simple flooding the request to the server which results in Denial of Service (DoS) [4].

In computing, the Denial of Service is to make a service provider to be unavailable to its intended users. Another variety of DoS, the Distributed Denial of Service (DDoS) [5-7] is attacks sent by two or more people where only one person will do attack in the case of DoS. Both the cases, the attackers aim is to prevent the legitimate users from accessing the services. The DoS and DDoS are the generic concepts which refer to any kind of client-server environment. In SOA, the DoS can be performed in a number of ways; one such of attack is reply attack. The replay attack is very easy to perform which does not require any technical expertise to interpret the request/response message in order to perform the attacks. The attacker is simply required to capture the request to the server and try to send the same message to the service provider with the intention to make the service provider busy so that its client will not be able to get its service in time. The attacker's goal is not to reveal any information from the messages, but to disturb the entire SOA and its operations; he/she can use Software to do the attack by sniffing the request packets to the server or can use the Hardware tools to perform the attacks. The replay attacks are very harm in nature which result in the entire web services down if they are not treated well by the security constraints in the server side, as the firewalls will not block them as they are legitimate requests in the firewall's point of view.

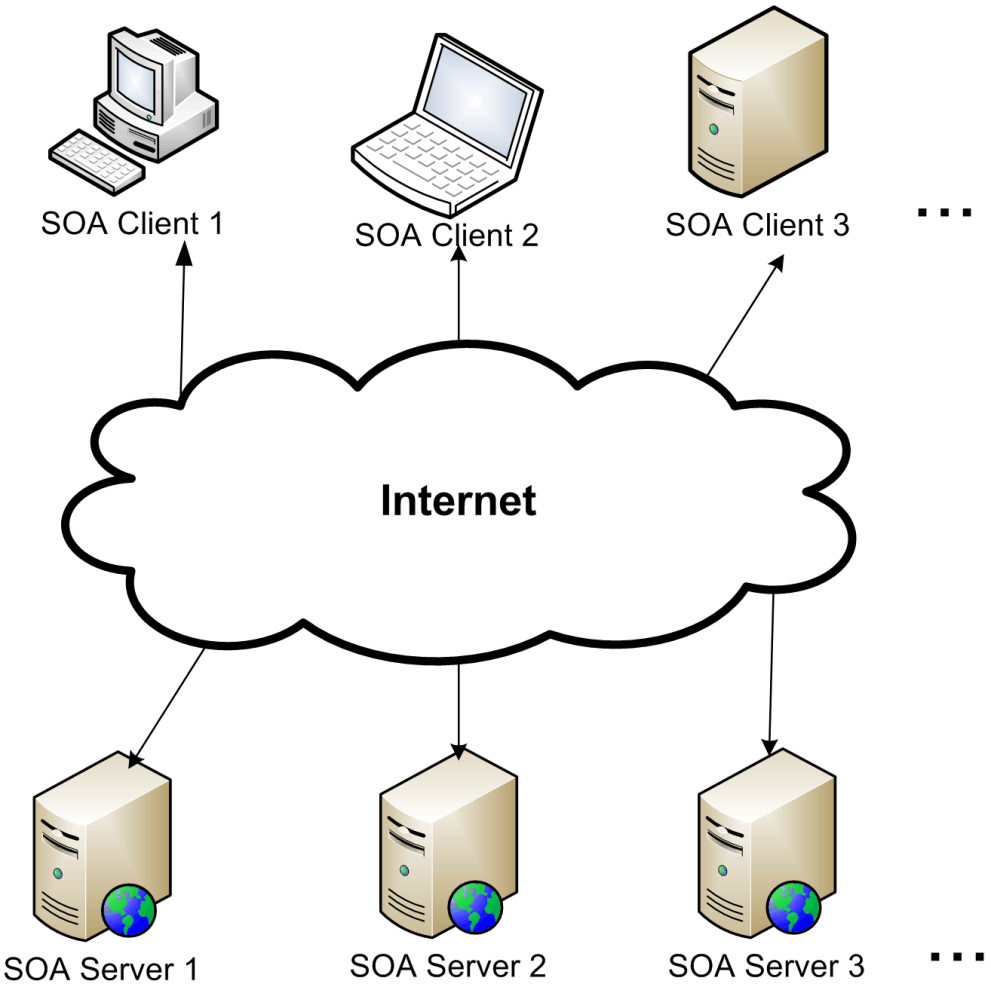

Fig. 1 SOA Client-Server Interaction

II. Replay Attacks and Their Severity

The Replay Attack is a kind of network attack in which the attacker retrieves the authenticated information by sniffing the data packets transmitted between server and client, and use this information to communicate with server as a legitimate user. Thus the attacker captures input SOAP messages, and sends it repetitively to the web service, which results in overloading of web service. For example, the genuine parties A and B are communicating over the network where A is the server and B is the client. The attacker C which eavesdrop the conversation between A and B, and use the retrieved information to prove as its valid identity

to A in order to make communicate with it. The attacker can issue the captured message request repetitively (replay) to the web service to overload it so that the web service will deny its service, resulting Denial of Service (DoS). A typical replay attack scenario is outlined in Fig. 2.

Replay vs Masquerading [8-10]: The masquerading (aka impersonation) is a variation of replay attacks where the attacker acts as if he is some other valid user who is communicating with the server, where in the replay attacks the attacker uses the same data packets which was captured during the communication of client to the server, results in reproducing effort.

Attacker's complexity: Replay attacks require small or no sophistication to perform.

Prerequisite to perform replay attacks: The attackers who try to do replay attacks should have the following two main prerequisites.

- Able to capture the SOAP request/response messages transferred between client and web service provider (web server)
- Able to reach the endpoint of the web service from the attacker's place, this part will become difficult if the web service provider and web service clients are within the secured network environment so that the web server can be reached by the users within the network.

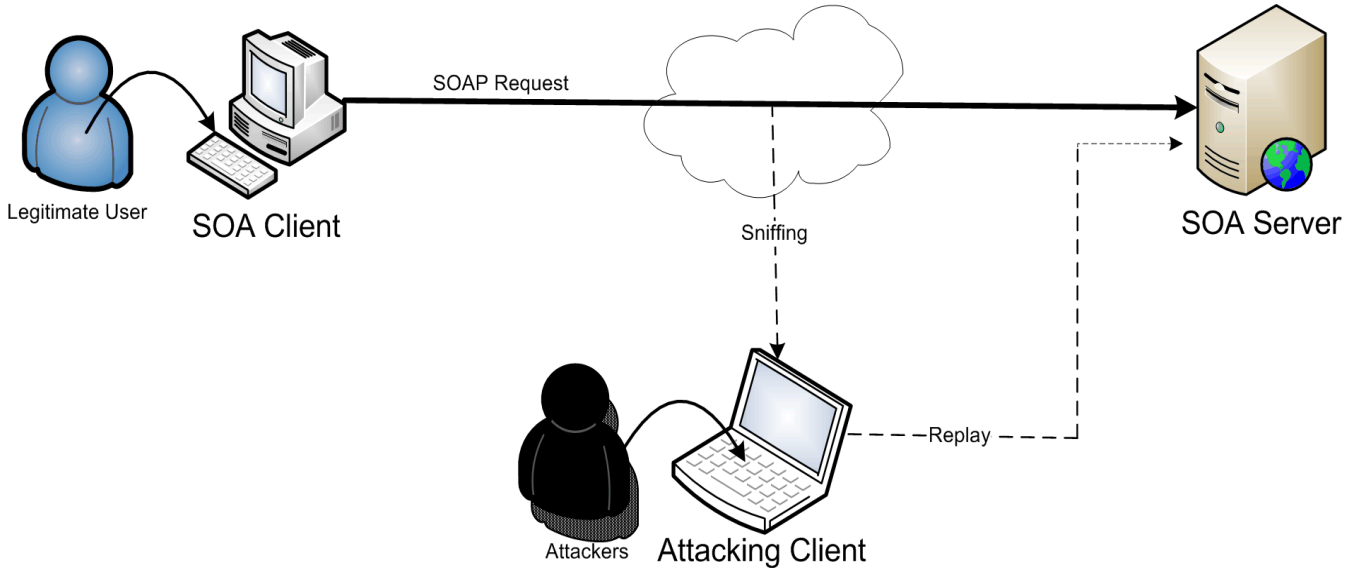

Fig. 1 Replay attack scenario in Web Service environment

Existing gaps in the available solutions: Reviewing the literature, we can see the solution for replay attacks are given in (i) Hardware level and (ii) Software level. The hardware products from vendors use network traffic information to prevent replay attacks; however this kind of filtration will not work for mission critical web services where the clients are accessing it very often such as stock exchange. Many researchers proposed security solutions for replay attacks, which includes secured sessions, encrypted messages, single time token access and hashing techniques, but none of them produce a comprehensive security solution for preventing replay attacks as those solutions work only for specific cases and heaving their own pros and cons [11]. For example, the solutions like encryption based and single time token needs the web services clients to do additional tasks to perform requests, which will breach SOA standard.

III. Proposed Security Solution for Preventing Replay Attacks

The proposed security solution finds out the previous request of time for the current request using the message identifier which is specified in SOAP request header. Then the previous time of the request is compared with the current time of the request and the time difference is greater than the defined guard time, then it is treated as no attack. If the time difference is less than or equals to the defined guard time, then it is considered for reply attack and the service is restricted for the request.

The algorithm used in the proposed security solution is given in Fig. 3 and its flow chart representation is outlined in Fig. 4. Whenever a SOAP request is given to the web service provider, this security solution checks whether it is a valid request or it is a request to do replay attack using message identifier for a particular web method call and timestamp of the current/previous request. If this security solution determines that the request is from attacker, then the request will be discarded and not allowed to serve for the request. This action may be notified to the client (suspected attacker) using the SOAP response.

The web service logging will improve the non-repudiation of the request/response. The request and response information will be maintained in an audit database so that the client cannot deny that they requested and in the same way the server cannot refuse that it did not respond. It is suggested to use Aspect Oriented Programming (AOP) for modern web service development and interceptors for the legacy code of web services.

```
Algorithm: Replay_Attack_Prevention
Input: SOAP Message Header
       Current Time of SOAP Request
Variable: MsgHdr: Request Message Header
          CurrTime: Current SOAP Request Time
          PrevTime: Last Time at which SOAP Request was made
Output:  Indicate attack status

Begin
  Step I:   /* Lookup Previous Time of the Request of same Message Id */
            PrevTime = Lookup(MsgHdr.MsgId)

  Step II:  /* Convert Current Request Time in seconds */
            CurrTimeSec = Call CalculateTimeInSeconds[CurrTime]

  Step III: /* Convert Previous Request Time in seconds */
            PrevTimeSec =  Call CalculateTimeInSeconds[PrevTime]

  Step IV: /* Check for occurrences of requests */
            If (CurrTimeSec – CurrTimeSec) > GuardTime Then
               Allow for the service
            Else
               Mark it as attack
            End If

  Step V:   /* Store the Current Time as Previous Time */
            CurrTime[MsgId] = PrevTime
End
```

Fig. 3 Algorithm for preventing replay attacks

IV. CONCLUSION

Replay attacks are simple to perform by attackers as they do not require any technical expertise to do; simply capture the legitimate request and try to keep on sending the captured request to the service provider to make it unavailable to its clients. In this paper, the replay attacks, their severity and existing gaps in the available solutions to these attacks are analyzed. A security algorithm for preventing these replay attacks is given with its high level flow of operations. This proposed solution is developed using Java technologies and testing using real time web services environment with logistic data.

References

1. Duggan, Dominic. "Service- Oriented Architecture." Enterprise Software Architecture and Design: Entities, Services, and Resources (2012): 207-358.
2. Komorita, Satoshi, et al. "Loosely coupled service composition for deployment of next generation service overlay networks." Communications Magazine, IEEE 50.1 (2012): 62-72.
3. Dua, Gagan, et al. "Replay Attack Prevention in Kerberos Authentication Protocol Using Triple Password." arXiv preprint arXiv:1304.3550 (2013).
4. Durcekova, Veronika, Ladislav Schwartz, and Nahid Shahmehri. "Sophisticated Denial of Service attacks aimed at application layer." ELEKTRO, 2012. IEEE, 2012.
5. Zargar, Saman Taghavi, James Joshi, and David Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." Communications Surveys & Tutorials, IEEE 15.4 (2013): 2046-2069.

6. Chung, Yoo. "Distributed denial of service is a scalability problem." ACM SIGCOMM Computer Communication Review 42.1 (2012): 69-71.
7. Gupta, B. B., Ramesh Chandra Joshi, and Manoj Misra. "Distributed Denial of Service prevention techniques." arXiv preprint arXiv:1208.3557 (2012).
8. Kaushik, Shant, and Sonia Sharma. "Securing Ad hoc Networks for Intrusion Detection, A study." (2015).
9. Sabahi, Farzad. "The Security of Vehicular Adhoc Networks." Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on. IEEE, 2011.
10. Khani, Payman, and Mehrdad S. Sharbaf. Investigation of Vehicular Networks and its Main Security Issues. No. 2014-01-0336. SAE Technical Paper, 2014.
11. Ahn, Jaw-Won, Jung-Ha Kang, and Eun-Gi Kim. "A Study on the Attack Prevention Methods for ARP Spoofing." International Information Institute (Tokyo). Information 17.11 (2014): 5443.

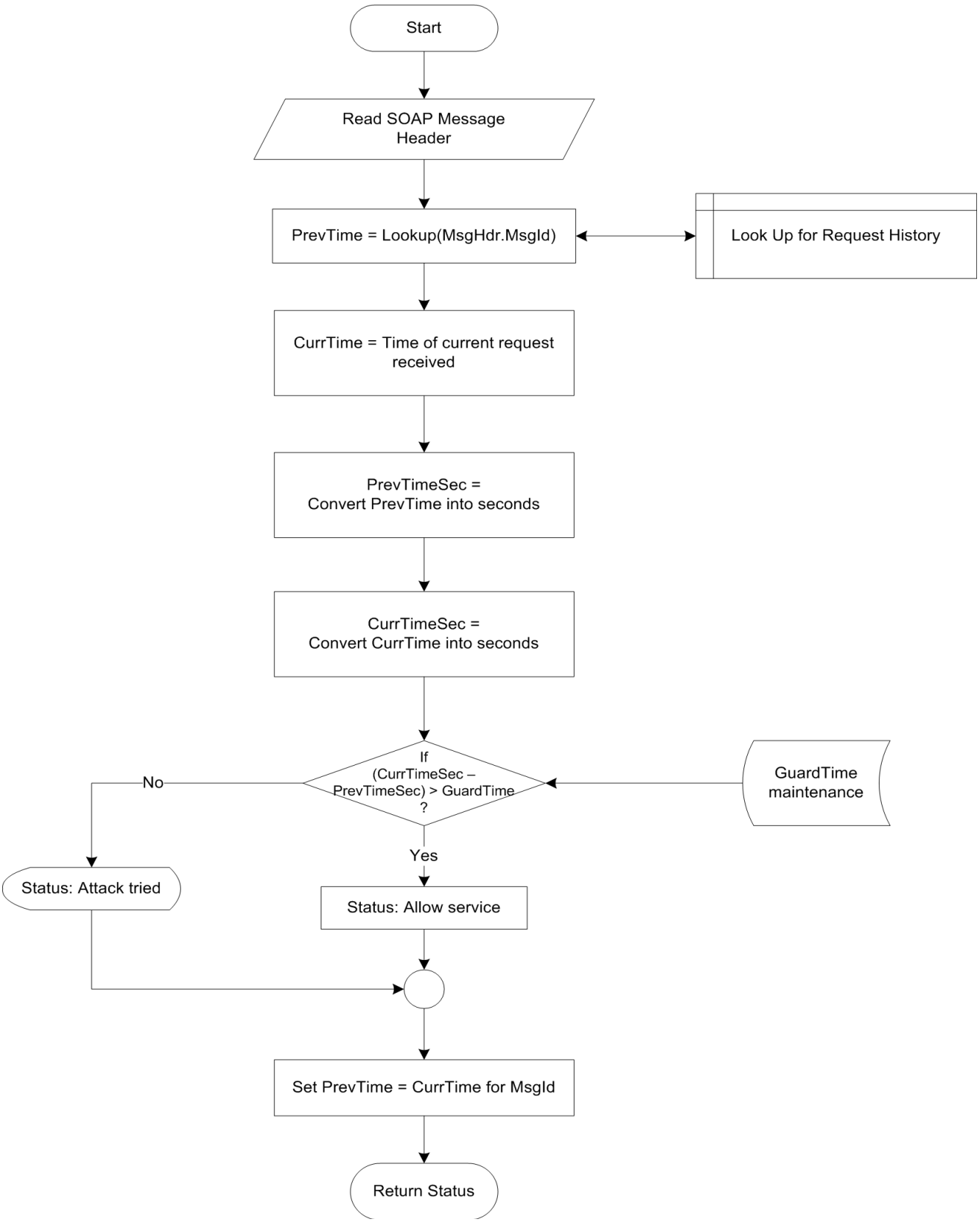

Fig. 4 Flow chart representation of the proposed security algorithm for preventing replay attacks

