Asian Journal of Information Science and Technology (AJIST)
Vol.4.No.1. 2016 pp 33- 38
ISSN: 2231-6108
available at: www.goniv.com
Paper Received :08-10-2016
Paper Accepted:22-12-2016
Paper Reviewed by: 1. R. Venkatakrishnan 2. Dr.G.Ramesh
Editor:Prof.P.Muthukumar

Self-Aware Message Validating Algorithm for Preventing XML Based Injection Attacks

A.Ramalakshmi
Asst. Professor
Department of Computer Science
Thiruvalluvar College
Papanasam, Vickramasingapuram
Tamilnadu-627425

ABSTRACT

Web Service attacks, generally called XML-based attacks, occur at the SOAP message level and thus they are not readily handled by existing security mechanisms in earlier firewalls. So as to provide robust security mechanisms for Web Services, XML filters have recently been introduced for Web Services security. In this research, a framework for dynamic XML filters are proposed, called self-aware message validating filter for XML based attacks, which supports detection and protection of XML-based attacks in real-time. A detailed design of the injection filter security model has been provided by validating schema information of the message with detection and protection policies**.** The information in the form of SOAP request is passed between client and server. Then it is processed in the Web Services opening an array of XML based injection attacks, for example, oversized message, message replay, parameter tampering, coercive parsing and semantic URL attacks. These attacks, among others, will be the focus of this section. Typical XML-based attacks include XSS injection attack, XPath injection attack, oversized message attack, replay attack, parameter tampering attack, XML injection attack, SQL injection attack and coercive parsing attack. For example, an XSS injection attack takes advantage of the weakness of CDATA of the parser of a service provider to allow malicious script in XML documents, forms or other methods in order to deform the information of the Website.

1.INTRODUCTION

An oversized message attack is a type of flooding attacks, where an attacker creates enormous level of traffic to a Web Service to exhaust its resources at the server side and parameter tampering attack can crash the server by sending unacceptable parameters. An XML-based attack can also be in a form of a distributed multi-faceted attack, Most of the XML-based attacks come in unpredicted format and so far the performance has not been studied carefully. For example, the attacker finds the XML tag for administrator then inserts that tag to act as high privileged user like administrator. This privilege violation would not be prevented the conventional firewall. Thereafter the attacker imitates the administrator's activities and gains all details of user realms.

Many security approaches have been developed for protecting Web Services, but they are vulnerable to predict and prevent the variety of attacks such as an XDoS attack. Some business concerns hesitate to adopt service oriented technologies because of lacking technology of robust security mechanism to prevent XML based attacks. An approach to defend against XML-based attacks at the application level is achieved in this work through self-aware message validating algorithm is proposed. It supports detection of XML-based attacks in real-time.In this research, the validation approach is used as a security mechanism and presented a framework for XML based injection filter. The architecture of the XML based injection filter service model is illustrated in Figure 1.1. As shown in the figure, an injection filter lies between service consumers and a service provider, and can be installed either on the same or a different machine where the actual Web Services are deployed. It interacts with service consumers through its User Interface (UI), which is responsible for receiving requests from and sending responses back to the login service.

1.1 ARCHITECTURE OF XML BASED INJECTION FILTER SERVICE

There are five major components supporting the filters, namely oversized message filter, message replay filter, parameter tampering filter, coercive parsing filter and semantic URL filter, which process the incoming request and state-based information, respectively. In the injection filter security model, input validation and protection are the major features for providing user access control, which ensure that only valid users are allowed to access certain Web Services.

1.2 Parameter Tampering Filter

The attacker adjusts the parameters in a SOAP message in an attempt to redirect the input validation in order to access unauthorized information. A change in integrity of the parameters is to detour the input validation and gains unauthorized access of some confidential functionality of Web Services. Because, the input parameters of an operation are given within a WSDL document, the hacker can play with different combinations of parameter patterns in order to access the unauthorized information. This filter checks the XML schema definition of received message for data type, null values. This filter checks the parameter for valid data. If it fails, then, it throws an error to the sender once. Even if the sender continues, his/her misbehaving with parameters leads to the disconnection of communication.

1.3 Coercive Parsing Filter

This filter verifies the namespaces and version mismatch received in the WSDL and SOAP files. This filtering policy used the fault values in SOAP fault code. The filter verifies the received message for wrong format of SOAP message by generating SOAP fault code. This filter blocks the input that has a strange format. This policy used the values in SOAP fault code. They are version mismatch and must understand fault code

1.4 Oversized Message Filter

The XML parsing of the service provider is directly affected by the size of the SOAP message. As a consequence, large amounts of Central Processing Unit (CPU) cycles are consumed when presented with large documents to process. A hacker can send a payload that is in alarming rate to exhaust systems resources. So, the filter is designed to refine the size of the message, requisition resources presented in the incoming message. Hence, the filter does the checking of three important parameters for the received SOAP message. First, it sets a request timeout to prevent infinite delay attacks. Then, it limits the amount of data that it retrieves.

1.5 Message Replay Filter

A hacker can resend the SOAP message requests to access the Web Service using other's login credentials. This kind of Web hacking will be escaped as a legitimate request because the source IP address is valid, the network packet attributes are valid and the HTTP request is well formed. Though, the business behavior of replay attacker is valid with unmatched parameters of the information is treated as an XML intrusion. Hence, the filter assigns an identifier for each incoming message and stored into the database to identify the replayed message. After that, the filter catches and matches the identifier of incoming messages and uses the replay detection policy to identify and reject messages which match an entry in the database of replay detection filter.

1.6 Semantic URL Filter

This is protected by giving token and timestamp for expiration. In this way, the filter is designed to protect the Web Service provider from the attack. The server and client shares the NONCE (Number used ONCE) for mutual identification. The random number generator is used the process identity of task to generate the nonce. Then, it is added with time stamp calculator to maintain the message expiration of the client/server. These parameters are shared either of the client/server to protect the integrity of the message.

2. ALGORITHM OF SELF-AWARE MESSAGE VALIDATION FOR XML BASED ATTACK INJECTION FILTER

A Web Service communicates with other applications over a network. There is no surety that the incoming message is requested from legitimate user, though the incoming request coming from authorized IP address. Meanwhile, the intruder includes some parameter to gain some data or to redirect the flow to some proprietary Web links. Based on the input information, the XML Request Handler can detect and verify XML based attack in real-time. The corresponding tables are created in the user information databases, which are used to store not only the current state and user information, but also the previous states and recent user information that are useful for attack detection and verification.

2.1 Detection of XML Based Injection Attacks

The XML request handler module is responsible for the dynamic detection and verification of the XML-based injection attacks by checking both the SOAP message and the parameters passed to a Web Service operation. The algorithm proposed by the XML request handler module is depicted in Figure 5.2 and explained how the input is treated as malformed input or not. As shown in the figure, when a SOAP message with a valid user request is sent to the XML request handler module, there the input is refined in all filters to verify the attack. This has been implemented by SAX parser of the server side filter to receive the legitimate schema of incoming message.

The process of detecting other types of XML-based attacks involves two major steps, which are detection of malformed SOAP messages and protection from attacks. Malformed SOAP messages are detected using the SOAP message validator. For example, to detect XML attacks, the handler module analyzes for possible flooding requests and keeps track of the allowable message size and the nesting depth in the incoming XML messages. If a certain type of attack is detected, the handler module will attempt to verify the attack using additional evidence from the blacklist database. Once an attack is confirmed, the SOAP message is rejected, and sent to the XML request handler module, where a rejection message is generated and sent back to the service consumer. In addition, the blacklist database is updated accordingly with the information related to the attack. Otherwise, the SOAP message is sent to the

deployed Web Service for service invocation, and after the service invocation, the results are sent back to the service consumer. The XML based attacks are categorized into five significant attacks and the protection mechanisms are implemented to refine and report the malformed input. First, the input filter is received in server side is passed to all filters to find the attack vectors. There, the input is passed to various input verification policies on parsers. Next, the captured input containing attacks are blocked and reported in XML request handler. Last, the legitimate input is forwarded to service provider.

2.3 Parameter tampering filter

In this, filter the received parameters are checked for data type, number of parameter and null values. This filter checks the parameter for valid data; if it fails, then, it throws an error to the sender once.

Even if the sender continues, his misbehaving with parameters leads to the disconnection of communication. To solve this problem, the proposed XSS filter was created with tamperchecker function as given in Figure 5.3. It checks the arguments for null values, data type, start element and end element of the received request from the client. The validation process itself is hidden from the client. The message validation of parameter tampering filter makes a number of checks to validate the message. That includes the verifying of the message payload is well-formed, means to verify whether the document follows all rules of recommended by W3C or not, also ensures to a predefined schema with acceptable data types and range of values. If the request succeeds all the validation checks that are performed by the message validator, then the service processes the message and forwards the request to the Web Service provider.

2.8 Oversized message filter

Denial of Service attacks happened by exhausting resources available in server side. Such attacks aim at reducing service availability by exhausting the resources of the service's host system, like memory, processing resources or network bandwidth. It is performed through query a service using a very large request message, which is called as over sized message. An oversized message attack is simple to perform, due to the high memory consumption of XML tags and its long processing duration. The total memory utilization to process one SOAP message is higher than the message size.

In the filter, the incoming message is checked for three important parameters. First, it sets a request timeout to prevent infinite delay attacks. Then, it limits the amount of data that it will retrieve. Last, it restricts the message from retrieving resources on the local host.The algorithm is depicts how it sets the values of attributes. The service provider had to set its maximum request length. The filter loaded the XML document and change XSD value maxMessageLength=1024 or any required buffer size. This compares the size of the request against the maximum allowable size that is specified for request messages.In the filter, the incoming message is checked for three important parameters. First, it sets a request timeout to prevent infinite delay attacks. Then, it limits the amount of data that it will retrieve. Last, it restricts the message from retrieving resources on the local host.

2.11 Message replay filter

An attacker attempt to resend SOAP requests to repeat sensitive transactions is called the message replay attack. Here, the client side message is assigned with an identifier and time stamp then send to the server for validation purpose. Thereafter, the filter captures the identifier of incoming messages and rejects messages that match an entry in the replay detection database. If the message identifier is valid because of its nonexistence, the filter compares the message timestamp to its clock time value for synchronization. If the message identifier has unacceptable identifier or any time stamp mismatch then, the message

is rejected. This can be done by calculating elapsedTime, cacheLifeSpan and maxMessagePeriod The time tolerance is the acceptable value time difference between the sender and the maximum message period is configured as 600 seconds. The filter calculates the elapsed time period of the message by deducting the created time value on the message from the present server time. For a message that appears have been created in the past or if the server and message creation times are equal will be rejected. Otherwise the message will be accepted only when its message age is less than or equal to the values for the maximum message period parameter and the elapsed time setting.

Cache Life Span = (MMP + ET * 2)

CLS - CacheLifeSpaninSeconds

MMP - MaximumMessagePeriod

ET - ElapsedTime

The algorithm of message replay filter is given in Figure 5.7. That accepts the XML document and retrieves its attribute through its parser. First it gets CLS and MMP attributes of incoming message. Then it calculates expiration to find that as an old message or new one. This is calculated by subtracting current time and time stamp of the received message. Then the difference value is assigned as the message period and stored into cache database. Next, it checks for messages where sender's clock is slower than the server's clock for first condition. Finally, it accounts for messages where the sender's clock is faster than the server's clock through the second condition. Then the identifier is stored into database. The unique identifier for the message is collected and saved in the replay finder cache before processing the request. The unique identifier is also used to solve the concurrency message collision issues. In scenario of two messages arrived at same time will be solved, if a second message arrives before the first message has finished executing. Some of the steps to be performed when attempting to detect replayed messages can harmfully affect system response time. For example, verifying the identifier of each incoming message and timestamp is computationally exhaustive will create XDoS.

Algorithm of message replay filter

3. Conclusion

The injection filter has been configured and embedded in administrator's. The administrator can enable or disable the filter based on their requirement. This feature will improve the speed of the server. The proposed system is compared with various existing systems namely input validator, AntiXSS filter, IE explorer, Opera and Firefox. The comparative analysis has been carried out with respect to number of attacks prevented.This filter protects Web Service attacks from intruders by verifying the attack from exception handler and throws exceptions to the client

Reference

1) Ahmad, K, Shekhar, J & Yadav, KP 2011, 'Coalesce Techniques to Secure net Applications and Databases against SQL Injection Attacks', Electronic Journal of engineering science and data Technology, vol. 3, no. 1, pp. 26-30.

2) Antunes, N & Vieira, M 2011, 'Enhancing Penetration Testing with Attack Signatures and Interface observance for the Detection of Injection Vulnerabilities in net Services', Proceedings of IEEE International Conference on Services Computing, pp. 104-111.

3) Antunes, N & Vieira, M 2012, 'Defending against net Application Vulnerabilities', IEEE laptop Society, vol. 45, no. 2, pp. 66-72.

4) Axelsson, S 2000, 'The Base-Rate misconception and also the problem Of Intrusion Detection', ACM Transactions on data and System Security (TISSEC), vol. 3, no. 3, pp. 186-205.
5) Bace, RG, 2000, 'Intrusion Detection', Macmillan Technical publication, Indianapolis, IN, USA.
6) Balzarotti, D, Cova, M, Felmetsger, V, Jovanovic, N, Kirda, E, Kruegel, C & Vigna, G 2008, 'Saner: Composing Static and Dynamic Analysis to Validate sanitisation in net Applications', Proceedings of the IEEE conference on Security and Privacy, pp. 387-401.
7) Bebawy, R, Sabry, H, El-Kassas, S, Hanna, Y & Youssef, Y 2005, 'Nedgty: net Services Firewall', Proceedings of the IEEE International Conference on net Services (ICWS'05), pp. 597- 601.
8) Bertino, E, Martino, L, Paci, F & Squicciarini, A 2010. 'Security for net Services and Service-Oriented Architectures', Springer house, Incorporated, first Edition, out there from: Springer, ISBN-10: 3540877894.
9) Bidou, R 2009, 'Attacks on net Services', OWASP, out there from :<https://www.owasp.org/images/6/6b/2009-05-06-OWASPFR-WebServices.pdf>. [20 Gregorian calendar month 2013].
10) Binbin Qu, Beihai Liang, Sheng Jiang & Chutian Ye 2013, 'Design of Automatic Vulnerability Detection System for net Application Program', continuing of Fourth IEEE International Conference on software system Engineering and repair Science (ICSESS), pp. 89-92.
11) Bisht, P., Sistla, AP., & Venkatakrishnan, VN 2010, 'TAPS: mechanically getting ready Safe SQL Queries', Proceedings of the seventeenth InternationalConference on laptop and Communications Security'2010, Chicago, USA, pp.645-647.
12) Boyd, SW, Kc, GS, Locasto, ME, Keromytis, AD & Prevelakis, V 2010, 'On the final relevance of Instruction-set Randomization', IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 3, pp. 255-270.
13) Capizzi, R, Longo, A, Venkatakrishnan, VN & Sistla, AP 2008, 'Preventing data Leaks Through Shadow Executions', In Proceedings of the pc Security Applications Conference IEEE, pp. 322-331.
14)Chang, CC & Lee, CY 2012, 'A Secure Single Sign-on Mechanism for Distributed laptop Networks', IEEE group action on Industrial physical science, vol.59, no.1, pp. 629-637.