# Adaptable Practices for Curbing XDoS Attacks

A. Karthigeyan, C. Andavar, A.Jaya Ramya

**Abstract**— An XDoS attack intends to exhaust the system resources of the server hosting a web service. An XDoS attack is done through SOAP messages. Cyber-criminals use distributed denial-of-service attacks (DDoS) and XML denial-of-service attacks (XDoS) to extort money from online service providers. This kind of attacks is normally targeted at a particular service provider to exhaust the network and system resources of the provider. This paper proposes a defense system against XDoS attacks by adapting some of the best practices for countering these XDoS attacks. The system is built on Web Services. It can be constructed and reconfigured easily by an attack victim.

**Index Terms**— Web Service, XDoS, DDoS, XML, SOAP

## 1 INTRODUCTION

DoS attacks were highly popular with the hacker community, and it's easy to understand. A single "script kiddie" attacker with a minimal amount of skill and resources could generate a flood of TCP SYN (for synchronize) requests sufficient to knock a site out of service. Over the years, SYN flood attacks have been largely mitigated by improvements in Web server software and network hardware. An XDoS attack is a content-borne attack whose purpose is to shut down a web service or system running that service. The 3 main strategies used in XDoS attacks are: Oversized payload, External entity references, Entity expansion XML DoS attacks are extremely asymmetric: to deliver the attack payload, an attacker needs to spend only a fraction of the processing power or bandwidth that the victim needs to spend to handle the payload. Worse still, DoS vulnerabilities in code that processes XML are also extremely widespread. Even if you're using thoroughly tested parsers, your code can still be vulnerable unless you take explicit steps to protect it.

The remainder of this paper is organized as follows. In the next section, we will introduce the basic concepts such as XML Web Service, Section III and IV describes Approaches and Limitation of Existing System. Section V discusses about Proposed Practices Adapted To Overcome XDoS Attacks and Architecture and finally, the paper concludes with the future work in Section VI.

## 2. Background

### 2.1 XML Web service

An XML Web service is a programmable entity that provides a particular element of functionality, such as application logic, and is accessible to any number of potentially disparate systems using ubiquitous Internet standards, such as XML
Detailed submission guidelines can be found on the author resources Web pages. Author resource guidelines are specific and HTTP. XML Web services depend heavily upon the broad acceptance of XML and other Internet standards to create an infrastructure that supports application interoperability at a level that solves many of the problems that previously hindered such attempts. An XML Web service can be used internally by a single application or exposed externally over the Internet for use by any number of applications. Because it is accessible through a standard interface, an XML Web service allows heterogeneous systems to work together as a single web of computation. Instead of pursuing the generic capabilities of code portability, XML Web services provide a viable solution for enabling data and system interoperability. XML Web services use XML-based messaging as a fundamental means of data communication to help bridge the differences that exist between systems that use incongruent component models, operating systems, and programming languages. Developers can create applications that weave together XML Web services from a variety of sources in much the same way that developers traditionally use components when creating a distributed application.

One of the core characteristics of an XML Web service is the high degree of abstraction that exists between the implementation and the consumption of a service. By using XML-based messaging as the mechanism by which the service is created and accessed, both the XML Web service client and the XML Web service provider are freed from needing any knowledge of each other beyond inputs, outputs, and location.

XML Web services are enabling a new era of distributed application development. It is no longer a matter of object model wars or programming language beauty contests. When systems are tightly coupled using proprietary infrastructures, this is done at the expense of application interoperability. XML Web services deliver interoperability on an entirely new level that negates such counterproductive rivalries. As the next revolutionary advancement of the Internet, XML Web services will become the fundamental structure that links together all computing devices.

- *A.Karthigeyan is currently working as a Senior Lecturer in the Department of CSE at Shree Motilal Kanhaiyalal Fomra Institute of Technology, affiliated to Anna University, Chennai. E-mail:akarthi78@yahoo.co.in*
- *C. Andavar is currently working as a Senior System Analyst in Ramco System, Chennai. E-mail:andavar.vnr@gmail.com*
- *A. Jaya Ramya is working as a software programmer in Mielsoft technologies, Puducherry. Email:ajramya.06@gmail.com*

## 3. Approaches of Existing System

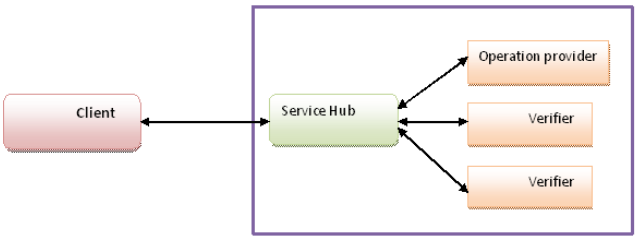
XDoS attacks intend to exhaust the victim's system resources and network bandwidth. The system resources can be exhausted by processing the requests sent in by the attacker. The attacker can exhaust the victim's network bandwidth by directing a large volume of traffic toward the victim's site. The scheme is avoids the network bandwidth of the Web Services providers being exhausted by hiding the locations of the Web Services providers from the public.

To avoid the system resources being exhausted by the attackers, the system carries out request message authentication and validation before the requests are processed by the Web Services providers. The scheme has two modes, i.e. the normal mode and the under-attack mode. An operations provider decides which mode the system works in. When an operations provider does not detect any attack activity, the system works in the normal mode. Otherwise, the system works in the under-attack mode. To minimize the delay in responding to users' requests, a service request is only authenticated and validated when the system works in the under-attack mode.

An operations provider subscribes to the service of a ServiceHub. The WSDL file describing the operations (i.e. services) provided by the operations provider binds the operations to the ServiceHub. Thus, the public perceive the operations as being hosted by the ServiceHub. As a result, all service requests are sent to the ServiceHub. Since an operations provider's address is unknown to the attackers, the attackers cannot send service requests directly to the operations provider.

Thus, the attackers cannot easily exhaust the network bandwidth of the operations provider. When working in the normal mode, clients' service requests do not need to be authenticated and validated. Figure 3.1(a) shows how the system works in the normal mode. Service requests are first sent to the ServiceHub (step 1). The ServiceHub forwards the requests to the operations provider (step 2). The operations provider sends the results back to the client through the ServiceHub (step 3 and 4).


a) Normal Mode




b) Under-attack Mode

**Figure 3.1 Client-Server Request/Response Operation**

In the under-attack mode, the service requests need to be authenticated and validated before being processed. The operations provider only processes a service request if the request can be successfully authenticated and validated. Therefore, the service provider does not waste system resources to process the attackers' requests. However, the authentication and validation mechanism also uses system resources. An attacker can still deplete the victim's system resources by sending in a large amount of requests that force the victim to authenticate and validate.

To counter this kind of attack, an operations provider, say op, subscribes the services provided by other service providers to delegate the authentication and validation task to the other ser-vice providers. The providers of the authentication and validation service are called verifiers. Since the authentication and validation is carried out by the verifiers, the attacker will not be able to exhaust op's system resources. Figure 3.1(b) shows how the system works in the under-attack mode.

The operations provider and the verifiers provide their services through the ServiceHub. They send and receive messages through the ServiceHub. Only the ServiceHub knows their IP addresses. Thus, they cannot exchange messages directly. In the under-attack mode, the operations provider informs the ServiceHub of the authentication and validation services that it sub-scribes to. The ServiceHub forwards the service requests to the corresponding verifiers. During the authentication process, the verifiers might need to exchange authentication information with the clients.

The ServiceHub is responsible for forwarding the messages ex-changed between the clients and the verifiers. If a service re-quest is authenticated and validated successfully, the verifier sends the service request to the operations provider through the ServiceHub. After processing the request, the operations provider sends the result back to the client through the ServiceHub.

## 4. Limitations of Existing System

1. Limiting the number of connections that a server will accept from a given IP address at any one time. Such a limit may help to prevent automated processes from exhausting the

server's resources
2. More strictly limiting the proposed restrictions depending on connection type, authentication type, or user class.
3. Less strict limits for server administrators compared to entities associated with registered accounts, and for entities associated with registered accounts compared to anonymous entities.
4. Less strict limits for entities that authenticate via strong authentication methods compared to entities that authenticate via weaker authentication methods
5. Less strict limits for connections made via the TCP binding compared to connections made via the HTTP binding

## 5. Proposed Practices Adapted To Overcome XDoS Attacks

The Proposed practices of XDoS Security in Web Server is implemented by means of following some of the best practices are given.

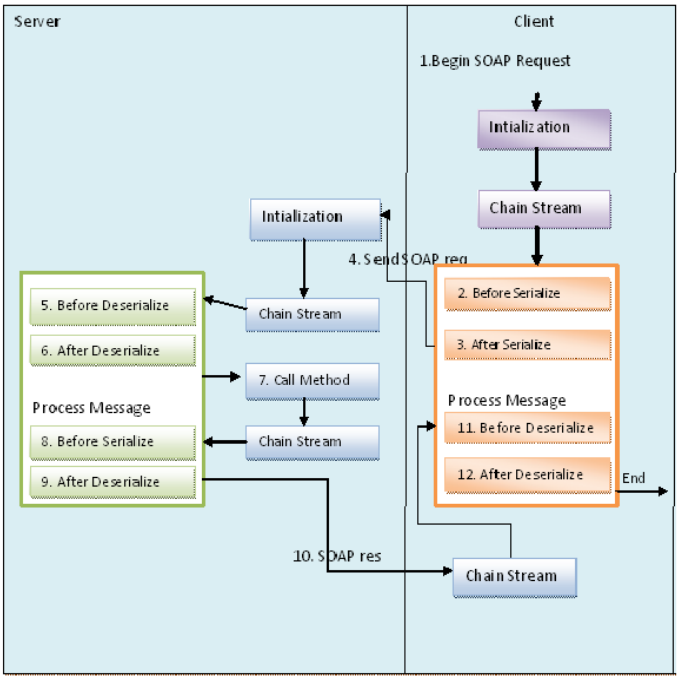
**Max Message Size**: Limit the size of payload to efficiently use CPU Cycles. When size exceeds discard the request/notify error <message size too big>
**Max Duration For A Host:** Limit the time to process a soap request. When time exceeds discard the request/notify time-out error.
**Request Rate From A Host**: Limit the maximum number of requests that can be received, in the interval period, from any one host. When time exceeds discard the request/notify limit exceeded.
**Block Interval**: The service will block access after one of the thresholds have been reached. The service will be available again once block interval is over.

**Override Parser Limits:**

(i) **XML Attribute Count**: Limits the number of attributes for any given element. Specify an integer.
(ii) **XML Bytes Scanned**: Limits the number of bytes contained in any given XML message. A value of 0 enforces no limit.
(iii) **XML Element Depth**: Limits the depth of nested elements in an XML message.
(iv) **XML Node Size**: Limits the size of any one XML node. The minimum allowed value is 1. The defined value can be larger than the value for the XML Bytes Scanned property. However, the value for the XML Bytes Scanned property takes precedence.

### 5.1 Proposed Architecture

The Proposed architecture describes the process of SOAP message. When a client invokes/calls a web service, it sends a request to the web service. This request is serialized into a SOAP message and sent over the network. On reaching the server side, this SOAP message is deserialized and the web service reads the request from the client. Depending on the client request, web service performs required operations and generates responses. This response is serialized into SOAP message at the server and deserialized at the client side. Similarly, the SOAP message is serialized at the server and deserialized at the client side when the response is sent from the server to the client. Thus the SOAP message goes through a process of serialization and deserialization both at the client and the server side. The various stages of SOAP messages are available in the SOAPMessageStage enumeration.

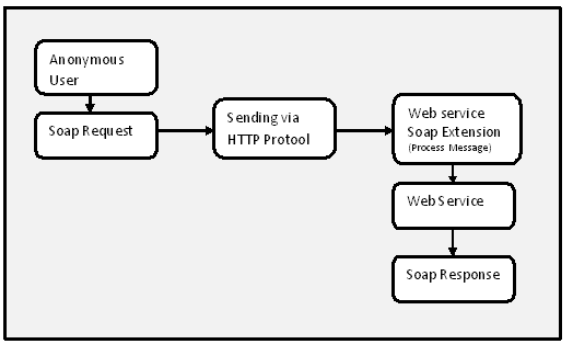



SOAP Extensions are components that can access the SOAP messages. Think of them as objects that sit on the HTTP pipeline who can pick the SOAP messages at each stage and manipulate them.

When the HTTP request comes from the client, it is handled by aspnet_isapi.dll. The appropriate handler for web services will be called and the web method will be invoked. It is during this stage where the SOAP Extension comes into picture. The SOAP Extension can access the SOAP message before and after calling the web method. Thus we now know in a vague manner what a SOAP extension is and where it fits in the life cycle of a SOAP message.

SOAP Extensions can be used for a number of purposes. They can be used to secure web services, compress the verbose SOAP messages, log messages etc.

### 5.2 Detailed Process Flow Structure

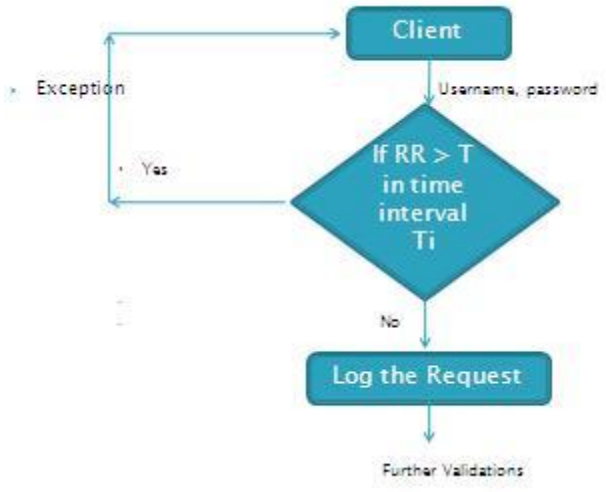



### 5.3 Workflow of the Proposed System

- The Client initiates a request to the server using a SOAP request.
- The request message is being deflated and then it is send to the server.
- Server side receives the deflated message from the client.
- The process of converting deflated message to inflated message takes place in the server side.
- Based on the client's request, the requested service is being invoked from the server.
- The invoked service on the server side is being deflated and it has been sent as response for the client request in the form of SOAP response.
- The client receives the response from the server.
- Again in the client end, the process of inflating takes place so as to view the response.

## 6. Implementation

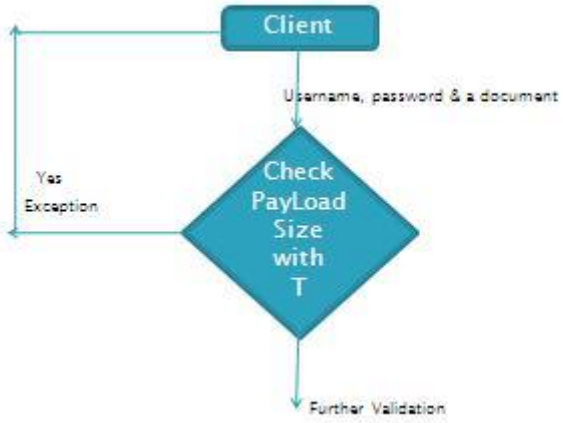
### 6.1 Request Rate From Host



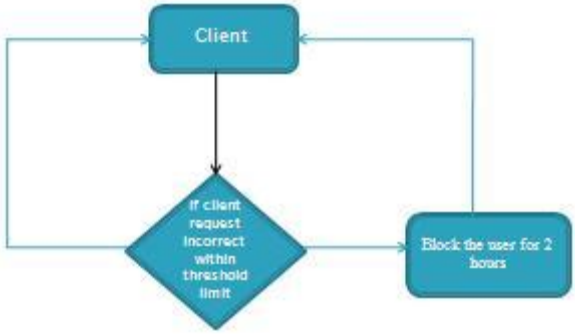

- Client submits the username and password.
- The client data is being stored in the form of a table which contains Username, Password, Login Time, client IP address and status are stored in the server machine .
- If a client request exceeds the threshold limit during the given time limit an exception is being raised stating that rate-of-request exceeded.
- On the other hand if not, the process the request.
- Note: Threshold value here means an optimal value. This will mentioned in all the other solutions

### 6.2 Oversized Payload

- Client sends a SOAP request which contains the XML file as an input.
- Calculate the size of the request file. If the size exceeds the threshold limit, then an exception is being raised stating that over sized payload.
- On the other hand if not, perform further processing.




### 6.3 Block Interval




- A client submits the username and password.
- On authentication if the client has found the username or password as incorrect, the client is given a maximum of 3 chances (threshold limit).
- If the threshold limit exceeds, the client account will

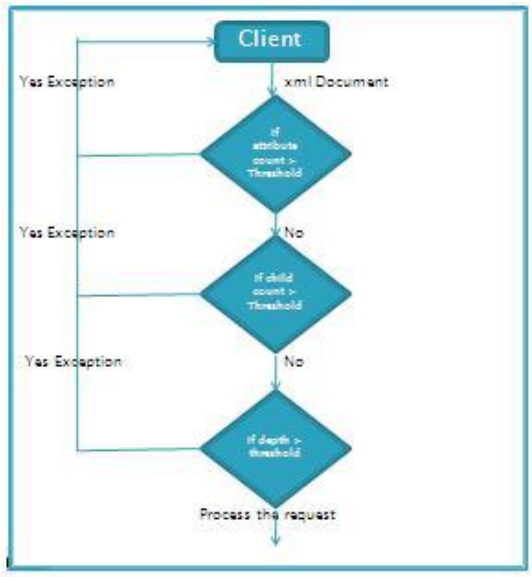
be pushed to the blocked state for two hours.

- After two hours, the client can be allowed to make another try.

### 6.4 Over-riding Parser Limits

- A client sends a SOAP request which contains an XML document
- On scanning the document, first the no. of attributes contained in it is compared with the threshold limit, if exceeded, an exception is raised.
- If the attribute count is within the threshold limit, then the child count is taken into account and it is compared within the limit, if exceeded and exception is raised

- On the other hand if the child count is within the limit, then the depth of the document is compared with the threshold limit, if exceeded, an exception is raised, else the request is processed.




## 7. Conclusion and Future Enhancement

This paper proposes a system adequate care has been taken and some of the best practices for countering the XDoS attacks has been implemented successfully. The performance of the system has been measured and it is found that faster detection allows the system to resist such attacks. Though the system has been developed by adapting the best practices, not all the practices has been adapted. The system can further be enhanced by following at-most best standards.

## References

[1] X. Ye, S. Singh. A SOA Approach to Counter DDoS Attacks, Proc. of the IEEE Intl. Conference on Web Services (ICWS'07), pp. 567-574, 2007

[2] S. Padmanabhuni, V. Singh, K. M. S. Kumar, and A. Chatterjee, Preventing Service Oriented Denial of Service (PreSODoS): A Proposed Approach, Proc. of the IEEE Intl. Conference on Web Services (ICWS'06), pp577 – 584, 2006

[3] R. Jaamour, XML security: Preventing XML bombs, http://searchsoftwarequality.techtarget.com/expert/Knowledg ebaseAnswer/0,289625,sid92_gci1168442,00.html

[4] B. Schneier, Applied Cryptography,Second Edition, John Wiley & Sons, 1996 T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. In Proc. Of Hotnets-II, Cambridge, MA, Nov. 2003

[5]M. Brambilla, S. Ceri, M. Passamani, A. Riccio: Managing Asynchronous Web Services Interactions, Proc. Of the IEEE Intl Conf on Web Services, 2004

[6] E. Kohler, Denial of Service Defense in Practice and Theory, USENIX'05, http://www.usenix.org/event/usenix05-/tech/slides/kohler.pdf

[7] R. Housley, W. Polk, W. Ford, and D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF RFC3280

[8] V. D. Gligor. Guaranteeing access in spite of distributed service-flooding attacks, in Proceedings of the Security Protocols Workshop, April 2003

[9] Alexandra Schäfer and Werner Rechert ,"Security and Web Services Specifications, Technologies and Frameworks" ,University of Technology Darmstadt Germany, 2002.

[11]Alex Stamos, "Attack for service : the Next generation Vulnerable Enterprice App." ,ISEC, Black HAT Briefings, bh-us-o5.pdf.

[12] (2006) Web service security. [Online]. Available: http://www.oasisopen. org/committees/wss/

[13] (2002) Xml-signature syntax and processing. [Online]. Available: http://www.w3.org/TR/xmldsig-core/

[14] M. McIntosh and P. Austel, "Xml signature element wrapping attacks and counterme

[15] asures," in SWS05: Proceedings of the 2005 workshop on Secure web services. New York, USA: ACM, 2005, pp. 20–27. [Online]. Available: http://doi.acm.org/10.1145/1103022.110302

[16] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Anarchitecture for mitigating DDoS attacks. Journal on SelectedAreas in Communications, 21(1):176–188, 2004

[17] W. Morein and A. Stavrou and D. Cook and A.Keromytis and V. Misra and D. Rubenstein, Using GraphicTuring Tests

to Counter Automated DDoS Attacks AgainstWeb Servers, Proc. of the 10th ACM Intl. Conf.

[18] S. Gajek, L. Liao, and J. Schwenk. Breaking and fixing the inline approach. In Proceedings of the 2007 ACM Workshop on Secure Web Services (SWS'07), pages 37–42, Fairfax, Virginia, USA, November 2007. Association for Computing Machinery.

[19] P. Grosso, E. Male, J. Marsh, and N.Walsh. Xpointer framework. W3C Recommendation, 2003.

[20] N. Gruschka. Protection Web Service by extended and efficient message validation. PhD thesis, University of Kiel, 2008.

[21] 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), pages 171–182, 2006.

[22] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WSSecurity 2004). 2006.

[23] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. In Proc. of Hotnets-II, Cambridge, MA, Nov. 2003